

(51) International Patent Classification:

G06K 9/46 (2006.01) G06T 1/40 (2006.01)
 G06N 3/02 (2006.01) G06T 11/60 (2006.01)
 G06N 3/08 (2006.01) G06T 7/10 (2017.01)

(21) International Application Number:

PCT/CA2019/050362

(22) International Filing Date:

25 March 2019 (25.03.2019)

(25) Filing Language:

English

(26) Publication Language:

English

(30) Priority Data:

62/646,934 23 March 2018 (23.03.2018) US
 62/783,251 21 December 2018 (21.12.2018) US

(71) Applicant: **THE GOVERNING COUNCIL OF THE UNIVERSITY OF TORONTO** [CA/CA]; Banting Institute, Room 413, 100 College St., Toronto, Ontario M5G 1L5 (CA).

(72) Inventors: **FIDLER, Sanja**; 476 Markham St., Toronto, Ontario M6G 2L3 (CA). **KAR, Amlan**; Suite 2514, 44 St. Joseph St., Toronto, Ontario M4Y 2W4 (CA). **LING, Huan**; Room 2204, 65 St. Mary St., Toronto, Ontario M5S 0A6 (CA). **GAO, Jun**; Room 2204, 65 St. Mary St., Toronto, Ontario M5S 0A6 (CA). **CHEN, Wenzheng**; 44 Gerrard

St. W., Toronto, Ontario M5G 2K2 (CA). **ACUNA MARRERO, David**; 402-30 Charles Street West, Toronto, Ontario M4Y 1R5 (CA).

(74) Agent: **HEER, Christopher**; 25 Adelaide Street East, Suite 1000, Toronto, Ontario M5C 3A1 (CA).

(81) Designated States (unless otherwise indicated, for every kind of national protection available): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States (unless otherwise indicated, for every kind of regional protection available): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM,

(54) Title: SYSTEMS AND METHODS FOR POLYGON OBJECT ANNOTATION AND A METHOD OF TRAINING AN OBJECT ANNOTATION SYSTEM

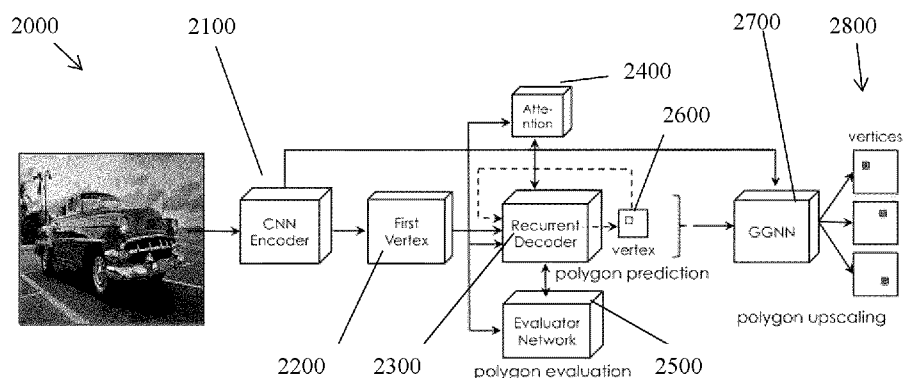


FIG. 2

(57) Abstract: The present invention relates generally to object annotation, specifically to polygonal annotations of objects. Described are methods of annotating an object including steps of receiving an image depicting an object, generating a set of image features using a CNN encoder implemented on one or more computers, and producing a polygon object annotation via a recurrent decoder or a Graph Neural Network. The recurrent decoder may include a recurrent neural network, a graph neural network or a gated graph neural network. A system for annotating an object and a method of training an object annotation system are also described.

TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW,
KM, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

SYSTEMS AND METHODS FOR POLYGON OBJECT ANNOTATION AND A METHOD OF TRAINING AN OBJECT ANNOTATION SYSTEM

FIELD OF THE INVENTION

[0001] The present specification relates generally to object annotation, specifically to polygonal annotations of objects.

BACKGROUND OF THE INVENTION

[0002] Detailed reasoning about structures or objects in images is helpful in numerous computer vision applications. For example, it is often critical in the domain of autonomous driving to localize and outline all cars, pedestrians, and miscellaneous static and dynamic objects. For mapping, there is often a need to obtain detailed footprints of buildings and roads from aerial or satellite imagery, while medical and healthcare domains often require automatic methods to precisely outline cells, tissues and other relevant structures.

[0003] Neural networks are sometimes an effective way of inferring semantic and object instance segmentation information in challenging imagery. Often, the amount and variety of data that the networks see during training drastically affects their performance at run time. Collecting ground truth instance masks, however, may be an extremely time-consuming task, such as requiring human annotators to spend 20-30 seconds per object in an image.

[0004] As object instance segmentation may be time consuming to annotate manually, several approaches seek to speed up this process using interactive techniques. In some approaches, scribbles are used to model the appearance of foreground and background, and segmentation is performed via graph-cuts. Some approaches use multiple scribbles on both the object and background, and have been used to annotate objects in videos.

[0005] In some approaches, scribbles are used to train convolutional neural networks ('CNN') for semantic image segmentation. In one approach, called GrabCut, 2D bounding boxes provided by an annotator are exploited, and pixel-wise foreground and background labeling is performed using expectation maximization ('EM'). In some approaches, GrabCut is combined with

convnets to annotate structures in imagery. In some approaches, pixel-wise segmentation of cars is performed by exploiting 3D point clouds inside user-provided 3D bounding boxes.

[0006] Many approaches to object instance segmentation operate on the pixel-level. Many rely on object detection, and use a convnet over a box proposal to perform the labeling. Although in some works, a polygon is produced around an object. Some approaches first detect boundary fragments, followed by finding an optimal cycle linking the boundaries into object regions. Some approaches produce superpixels in the form of small polygons which are further combined into an object.

[0007] In some approaches, polygon object representation has been introduced as an alternative to labeling each individual pixel. One benefit of polygon object representation is that it is sparse; only a few vertices of a polygon represent large image regions. For example, this may allow the user to easily introduce any correction, by correcting the wrong vertices. A recurrent neural network ('RNN') may further provide a strong model as it captures non-linear representation of shape, thus effectively capturing typical shapes of objects. This may be particularly important in ambiguous cases such as imagery containing shadows and saturation.

[0008] For example, Polygon-RNN is a conceptual model for semi-automatic and interactive labeling to help speed up object annotation. Instead of producing pixel-wise segmentation of an object, as is done in some interactive tools such as Grabcut, Polygon-RNN predicts the vertices of a polygon that outlines the object. Polygon representation may provide several benefits; it is sparse with only a few vertices representing regions with a large number of pixels, it may be easier for an annotator to interact with, and the model may be able to directly take annotator inputs to re-predict a better polygon that is constrained by the corrections. In some embodiments, polygon representation models have shown high annotation speed-ups on autonomous driving datasets.

[0009] Further improved polygon representation models may further speed up annotation time, improve neural network learning from polygon representation models, and increase the output resolution of polygons.

SUMMARY OF THE INVENTION

[0010] In an embodiment of the present invention, there is provided a method of annotating an object, comprising: receiving an image depicting an object; generating a set of image features using a CNN encoder implemented on one or more computers; predicting a set of first vertex predictions using the set of image features; producing a set of polygon representations of the object using a recurrent decoder that exploits the set of first vertex predictions and the set of image features, the recurrent decoder including a RNN implemented on one or more computers; and selecting a polygon object annotation from the set of polygon representations using an evaluator network.

[0011] In an embodiment of the present invention, there is provided a system for object annotation, comprising: a CNN encoder implemented by one or more computers for generating image features from a received image, the image features for predicting a first vertex of an object annotation; a recurrent decoder for generating a set of polygon representations of an object in the received image, the recurrent decoder including a RNN implemented by one or more computers; and an evaluator network for selecting a polygon object annotation from the set of polygon representations.

[0012] In an embodiment of the present invention, there is provided a method of training an object annotation system having a CNN encoder, an RNN recurrent decoder and an evaluator network, comprising: receiving a training dataset; initiating a training sequence for setting one or more weight matrices of the object annotation system using managed learning environment training, such as maximum likelihood learning; and fine-tuning the one or more weight matrices of the object annotation system using reinforcement learning to produce a trained object annotation system.

[0013] In a further embodiment of the present invention, there is provided a method of annotating an object, comprising: receiving an image depicting an object, the image comprising an n-dimensional array-like data structure; generating a set of image features using a CNN encoder implemented on one or more computers; initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image; predicting a location shift for each node simultaneously using a Graph Neural Network (GNN); iterating predictions through the GNN for each node, each iteration defining a new location shift

for each node based on node locations for each node from the previous iteration; and producing an object annotation based on a final iteration; wherein the object is parametrized with one of polygons and splines.

[0014] In a further embodiment of the present invention, there is provided a system for object annotation, comprising: a CNN encoder implemented by one or more computers for generating a set of image features from a received image, the image comprising an n-dimensional array-like data structure, and for initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image; a Graph Neural Network (GNN) implemented by one or more computers for predicting a location shift for each node simultaneously and iterating predictions through the GNN for each node, each iteration defining a new location shift for each node based on node locations for each node from the previous iteration; and an output selector for producing an output based on a final iteration from the GNN.

[0015] In a further embodiment of the present invention, there is provided a non-transient computer-readable medium comprising instructions for a method of annotating an object, the method comprising: receiving an image depicting an object, the image comprising an n-dimensional array-like data structure; generating a set of image features using a CNN encoder implemented on one or more computers; predicting a set of first vertex predictions using the set of image features; producing a set of polygon predictions of the object using a recurrent decoder that exploits the set of first vertex predictions and the set of image features, the recurrent decoder implemented on one or more computers; and selecting a polygon object annotation from the set of polygon predictions.

[0016] In a further embodiment of the present invention, there is provided a non-transient computer-readable medium comprising instructions for a method of annotating an object, the method comprising: receiving an image depicting an object, the image comprising an n-dimensional array-like data structure; generating a set of image features using a CNN encoder implemented on one or more computers; initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image; predicting a location shift for each node simultaneously using a Graph Neural Network (GNN); iterating predictions through the GNN for each node, each iteration defining a new location shift

for each node based on node locations for each node from the previous iteration; and producing an object annotation based on a final iteration; wherein the object is parametrized with one of polygons and splines.

[0017] In a yet further embodiment of the present invention, there is provided a non-transient computer-readable medium comprising instructions for a method of training an object annotation system having a CNN encoder, an RNN recurrent decoder and an evaluator network, the method comprising: receiving a training dataset; initiating a training sequence for setting one or more weight matrices of the object annotation system using managed learning environment training; and fine-tuning the one or more weight matrices of the object annotation system to produce a trained object annotation system.

BRIEF DESCRIPTION OF THE FIGURES

[0018] The principles of the invention may better be understood with reference to the accompanying figures provided by way of illustration of an exemplary embodiment, or embodiments, incorporating principles and aspects of the present invention, and in which:

[0019] FIG. 1 shows examples of four dataset images;

[0020] FIG. 2 shows a schematic diagram of aspects of an embodiment;

[0021] FIG. 3 shows a schematic diagram of aspects of an embodiment;

[0022] FIG. 4 shows a schematic diagram of aspects of an embodiment;

[0023] FIG. 5 shows a schematic diagram of aspects of an embodiment;

[0024] FIG. 6 shows a representation of experimental results of an embodiment;

[0025] FIG. 7 shows a representation of experimental results of an embodiment;

[0026] FIG. 8 shows a representation of experimental results of an embodiment;

[0027] FIGs. 9A to 9D show representations of experimental results of an embodiment;

[0028] FIG. 10 shows examples of images annotated with an embodiment;

- [0029] FIG. 11 shows examples of images annotated with embodiments;
- [0030] FIG. 12 shows examples of images annotated with an embodiment;
- [0031] FIG. 13 shows examples of images annotated with an embodiment;
- [0032] FIG. 14 shows a flow diagram of an embodiment;
- [0033] FIG. 15 shows a schematic diagram of an embodiment;
- [0034] FIG. 16 shows a flow diagram of an embodiment;
- [0035] FIG. 17 shows a schematic diagram of aspects of an embodiment;
- [0036] FIG. 18 shows a representation of an aspect of an embodiment;
- [0037] FIG. 19 shows a schematic diagram of aspects of an embodiment;
- [0038] FIG. 20 shows examples of images annotated with an embodiment;
- [0039] FIG. 21 shows examples of images annotated with an embodiment;
- [0040] FIG. 22 shows examples of images annotated with an embodiment;
- [0041] FIG. 23A and FIG. 23B show a representation of experimental results of an embodiment;
- [0042] FIG. 24 shows a representation of experimental results of an embodiment;
- [0043] FIG. 25 shows examples of images annotated with an embodiment;
- [0044] FIG. 26 shows examples of images annotated with an embodiment;
- [0045] FIG. 27 shows a method of the present invention according to an alternative embodiment; and
- [0046] FIG. 28 shows a system of the present invention according to an alternative embodiment.

DETAILED DESCRIPTION OF EMBODIMENTS OF THE INVENTION

[0047] The description that follows, and the embodiments described therein, are provided by way of illustration of an example, or examples, of particular embodiments of the principles of the present invention. These examples are provided for the purposes of explanation, and not of limitation, of those principles and of the invention. In the description, like parts are marked throughout the specification and the drawings with the same respective reference numerals. The drawings are not necessarily to scale and in some instances proportions may have been exaggerated in order to more clearly to depict certain features of the invention.

[0048] This description relates to improvements in polygon representation models for object recognition, such as improvements to the Polygon-RNN model disclosed in D1 (*L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a polygon-rnn. In CVPR, 2017*), the entirety of which is hereby incorporated by reference. In particular, it relates to changes to the neural network architecture, a new learning algorithm to train polygon models using reinforcement learning, and increasing the output resolution of the polygon using a Graph Neural Network. This description also relates to the robustness of polygon models with respect to noise, and their generalization capabilities to out-of-domain imagery.

[0049] The present description relates to a fully automatic mode, in which an annotator is not in the loop, and a partially automatic mode, in which an annotator is in the loop. In a fully automatic mode, with no annotator in the loop, changes disclosed herein to existing polygon models may result in improved Intersection over Union (IoU). In an interactive mode, with an annotator in the loop, changes to existing polygon models may allow for significantly less human annotator clicks.

[0050] The present description relates to online fine-tuning to achieve a higher annotation speed-up, such as on out-of-domain dataset annotation.

[0051] As shown at 1000 in FIG. 1, a polygon object recognition model may be applied to a variety of dataset domains, including autonomous driving imagery, medical imagery, and aerial imagery, and may also be applied to other general scenes. In this sense, an image is n-dimensional array-like data structure provided as an input to the system. This input may be received as sensor

data in the appropriate format, and known image types that may be input include RGB images, thermal images, depth images and hyperspectral images.

[0052] 1. The Present Model

[0053] An embodiment of the present application is an improved version of the Polygon-RNN model disclosed in D1; Polygon-RNN++. Polygon-RNN++ expects an annotator to provide a bounding box around an object of interest in a figure. The polygon object recognition model then extracts an image crop enclosed by an enlarged box, where the enlarged box is the annotator-provided bounding box enlarged by 15%.

[0054] In some embodiments, the polygon model exploits a CNN and RNN architecture, with a CNN serving as an image feature extractor, and the RNN decoding one polygon vertex at a time. Output vertices may be represented as a location in a grid.

[0055] In the embodiment depicted in schematic form in FIG. 2, an encoder 2100 generates image features that are used to predict the first vertex 2200. The first vertex 2200 and the image features are used by the recurrent decoder 2300. This uses visual attention 2400 at each time step to produce a polygon one vertex 2600 at a time. A learnt evaluator network 2500 selects the best polygon out of candidates proposed by the decoder 2300. Finally, a graph neural network 2700 generates polygons at a higher resolution, defined by vertices 2800.

[0056] This model naturally incorporates a human in the loop, allowing the annotator to correct an erroneously predicted vertex. This vertex is then fed back to the model, helping the model to correct its prediction at the next time steps.

[0057] 1.1. Residual Encoder with Skip Connections

[0058] Many networks perform repeated down-sampling operations at consecutive layers of a CNN, which impacts the effective output resolution in tasks such as image segmentation. However, in some embodiments, such as architecture 3000 depicted in FIG. 3, a ResNet-50 architecture is modified by reducing the stride of the network and introducing dilation factors, which allows for an increase in the resolution of the output feature map without reducing the

receptive field of individual neurons. The original average pooling and fiber channel ('FC') layers may also be removed.

[0059] Some embodiments of the present invention further include a skip-layer architecture which aims to capture both low-level details, such as edges and corners, as well as high-level semantic information. In some models down-sampling is performed in the skip-layer architecture, built on top of Visual Geometry Group ('VGG'), before concatenating the features from different layers. However, in embodiments of the present invention, all the outputs of the skip layers are concatenated at the highest possible resolution, and a combination of conv layers and max-pooling operations are used to obtain the final feature map. For example, conv filters with a kernel size of 3×3 , batch normalization and ReLU (rectified non-linear units) non-linearities may be employed. In cases where the skip-connections have different spatial dimensions, bilinear upsampling may be used before concatenation. Representative architecture 3000 is depicted in FIG. 3, wherein the final feature map is referred to as the skip features 3200.

[0060] 1.2. Recurrent Decoder

[0061] As in D1, embodiments of the present application use a Recurrent Neural Network to model the sequence of 2D vertices of the polygon outlining an object. In some embodiments, Convolutional Long Short-Term Memory ('LSTM') is also used to preserve spatial information and to reduce the number of parameters to be learned.

[0062] Embodiments of the present application use a two-layer ConvLSTM with a 3 by 3 kernel with 64 and 16 channels, respectively. Batch norm is applied at each time step, without sharing mean or variance estimates across time steps. Output at time step t is represented as one-hot encoding of $(D \times D) + 1$ elements, where D is the resolution predicted. For example, D may be set to 28. The first $D \times D$ dimensions represent the possible vertex positions and the last dimension corresponds to the end-of-seq token that signals that the polygon is closed.

[0063] Attention Weighted Features: A mechanism akin to attention may be exploited in an RNN. For example, at time step t the weighted feature map may be computed as in equation (1), where $\circ$ is the Hadamard product, x is the skip feature tensor, and $h_{1,t}$ and $h_{2,t}$ are the hidden state tensors from the two-layer ConvLSTM. f_1 and f_2 map $h_{1,t}$ and $h_{2,t}$ to $R^{D \times D \times 128}$ using one

fully-connected layer. f_{att} takes the sum of its inputs and maps it to $D \times D$ through a fully connected layer, giving one “attention” weight per location.

$$\begin{aligned}\alpha_t &= \text{softmax}(f_{att}(x, f_1(h_{1,t-1}), f_2(h_{2,t-1}))) \\ F_t &= x \circ \alpha_t\end{aligned}\tag{1}$$

[0064] In some embodiments, the previous RNN hidden state is used to gate certain locations in the image feature map, allowing the RNN to focus only on the relevant information in the next time step. The gated feature map F_t is then concatenated with one-hot encodings of the two previous vertices y_{t-1} , y_{t-2} and the first vertex y_0 , and passed to the RNN at time step t .

[0065] First Vertex: In some embodiments, given a previous vertex and an implicit direction, the next vertex of a polygon is always uniquely defined, except for the first vertex. The first vertex may be treated as a special case. In some embodiments it may be predicted using an additional architecture, trained separately. In some embodiments, another branch may be added from the skip-layer architecture, constituting of two layers, each of dimensions $D \times D$, the first layer predicting edges, while the second predicts the vertices of the polygon, with the first vertex sampled from the final layer of this branch.

[0066] 1.3. Training Using Reinforcement Learning

[0067] Training a model using the cross-entropy loss at each time step may have two limitations; managed learning environment (‘MLE’) over-penalizes the model (for example when the predicted vertex is on an edge of the GT polygon but is not one of the GT vertices), and it optimizes a metric that is very different from the final evaluation metric (i.e. IoU). Training a model using ‘teacher forcing’ in which a model is trained following a typical training regime where the GT vertex is fed to the next time step instead of the model’s prediction, may create a mismatch between training and testing known as the exposure bias problem.

[0068] While such training techniques could be used in some embodiments, in other embodiments MLR training may only be used as an initialization stage. The polygon prediction task is reformulated as a reinforcement learning problem and the network is fine-tuned using reinforcement learning (‘RL’). During this phase, the network is allowed to discover policies that

optimize the desirable, yet non-differentiable evaluation metric (IoU) while also exposing it to its own predictions during training.

[0069] 1.3.1 Problem Formulation

[0070] In embodiments, the recurrent decoder is viewed as a sequential decision making agent. The parameters Θ of the encoder-decoder architecture define its policy p_θ for the selection of the next vertex v_t . At the end of the sequence, the agent observes a reward r . The reward r is computed as the IoU between the mask enclosed by the generated polygon and the ground-truth mask. To maximize the expected reward, our loss function becomes function (2).

$$L(\theta) = -\mathbb{E}_{v^s \sim p_\theta} [r(v^s, m)] \quad (2)$$

Where $v^s = (v_1^s, \dots, v_T^s)$, and v_t^s is the vertex sampled from the model at time t . Here, m is the ground truth mask for the given object instance and $r = \text{IoU}(\text{mask}(v^s), m)$.

[0071] 1.3.2. Self-Critical Training with Policy Gradients

[0072] Some embodiments using the REINFORCE approach to compute the gradients of the expectation result in function (3).

$$\nabla L(\theta) = -\mathbb{E}_{v^s \sim p_\theta} [r(v^s, m) \nabla \log p_\theta(v^s)] \quad (3)$$

[0073] Some embodiments use Monte-Carlo sampling with a single sample to compute the expected gradient. This approach may exhibit high variance and may be highly unstable without proper context-dependent normalization. In some embodiments a learned baseline may be used, and may be subtracted from the reward. In some embodiments the self-critical method may followed, and the test-time inference reward of the model is used as the baseline. Accordingly, the gradient of the loss function may be reformulated as function (4).

$$\nabla L(\theta) = -[(r(v_s, m) - r(\hat{v}_s, m))\nabla \log p_\theta(v_s)] \quad (4)$$

Where $r(\hat{v}_s, m)$ is the reward obtained by the model using greedy decoding.

[0074] To control the level of randomness in the vertices explored by the model, in some embodiments a temperature parameter τ is introduced in the softmax operation that gets the probability distribution of the policy. This ensures that the sampled vertices lead to well-behaved polygons. For example, τ may be set to 0.6.

[0075] 1.4. Evaluator Network

[0076] A well-chosen first vertex may be important as it biases the initial predictions of the RNN, when the model does not have a strong history to reason about the object to annotate. This may be particularly important in cases of occluding objects. It may be desirable for the first vertex to be far from the occlusion boundaries so that the model follows the object of interest. In RNNs, beam search may be used to prune off improbable sequences (such as when the model starts to follow an occluding object). However, since classical beam search uses log probabilities to evaluate beams, it may not directly apply to a model which aims to optimize IoU. A point on an occlusion boundary may exhibit a strong edge and thus may have a high log probability during prediction, reducing the chances of it being pruned by beam search.

[0077] A solution to this problem may be to use an evaluator network at inference time, aiming to effectively choose among multiple candidate polygons. An evaluator network takes as input the skip features, the last state tensor of the ConvLSTM, and the predicted polygon, and tries to estimate its quality by predicting its IoU with Gamma Testing ('GT'). The network may have two 3×3 convolutional layers followed by a FC layer, forming another branch in the model. The architecture of an embodiment evaluator network is depicted in FIG. 4. While the full model may be trained end-to-end during the RL step, in some embodiments the evaluator network may be trained separately after the RL fine-tuning has converged.

[0078] During training, the mean squared error of function (5) may be minimized.

$$L = [p - \text{IoU}(m_{v_s}, m)]^2 \quad (5)$$

Where p is the network's predicted IoU, m_{v_s} is the mask for the sampled vertices and m is the ground-truth mask. To ensure diversity in the vertices seen, polygons may be sampled with $\tau = 0.3$. In embodiments, this network is not used as a baseline estimator during the RL training step as the self-critical method may produce better results.

[0079] Inference: At test time, K top scoring first vertex predictions may be taken. For each of these, polygons are generated via classical beam-search (using log probability). This yields K different polygons, one for each first vertex candidate. The evaluator network may be used to choose the best polygon. For example, $K = 5$ may be used. While one could use the evaluator network instead of beam-search at each time step, this may lead to impractically long inference times. For example, it may be desired to run a model at 36ms per object instance on a Titan XP.

[0080] Annotator in the Loop: Where an annotator is in the loop, the annotator may correct the vertices in sequential order. Each correction may then be fed back to the model, which may re-predict the rest of the polygon.

[0081] 1.5. Upscaling with a Graph Neural Network

[0082] The model disclosed above may produce polygons at a resolution of $D \times D$. For example, D may be set to 28 to satisfy memory bounds and to keep the cardinality of the output space amenable. In other embodiments, a Gated Graph Neural Network (GGNN) may be used, in order to generate polygons at a much higher resolution. When training the RNN decoder, the GT polygons may be simplified at their target resolution (co-linear vertices are removed) to alleviate the ambiguity of the prediction task. Thus, at a higher resolution, the object may have additional corners (vertices), effectively changing the topology of the polygon.

[0083] Some embodiments build on top of the sequence of polygon vertices generated by the RNN decoder. These vertices are treated as nodes in a (cycle) graph. To account for the change in geometry at a higher resolution, a node is added in between two consecutive nodes, with its location being in the middle of their corresponding edge. The last and the first vertex are also

connected, effectively converting the sequence into a cycle. Neighboring nodes are connected using 3 different types of edges, as shown in FIG. 5. GGNN then defines a propagation model that extends RNNs to arbitrary graphs, effectively propagating information between nodes, before producing an output at each node. Here, the aim is to predict the relative offset of each node (vertex) at a higher resolution. As such, GGNN allows for an improvement in predictions, both for the initial vertices of the polygon where the RNNs history took less effect, as well as effectively upscaling old and new vertices. The model is visualized in FIG. 5.

[0084] Gated Graph Neural Network: For completeness, the GGNN model is summarized. GGNN uses a graph $\{V, E\}$, where V and E are the sets of nodes and edges, respectively. It includes a propagation model performing message passing in the graph, and an output model for prediction tasks. The initial state of a node v is represented as x_v and the hidden state of node v at time step t as h_v^t . The basic recurrence of the propagation model is set out in function (6).

$$\begin{aligned} h_v^0 &= [x_v^\top, 0]^\top \\ a_v^t &= A_v^\top [h_1^{t-1^\top}, \dots, h_{|V|}^{t-1^\top}]^\top + b \\ h_v^t &= f_{GRU}(h_v^{t-1}, a_v^t) \end{aligned} \quad (6)$$

Where the matrix $A \in R^{|V| \times 2N|V|}$ determines how the nodes in the graph communicate with each other, where N represents the number of different edge types. Messages are propagated for T steps. The output for node v is then defined as in function (7).

$$\begin{aligned} h_v &= \tan(f_1(h_v^T)) \\ \text{out}_v &= f_2(h_v) \end{aligned} \quad (7)$$

Where f_1 and f_2 are MLP, and out_v is v 's desired output.

[0085] PolygonRNN++ with GGNN: To obtain initial observations for the GGNN model, another branch may be added on top of the skip-layer architecture, in particular from the $112 \times 112 \times 256$ feature map, such as the concat feature map of FIG. 4. A conv layer with 256 filters of size 15×15 may be exploited, giving a feature map of size $112 \times 112 \times 256$. For each node v in the graph, a $S \times S$ patch around the scaled (v_x, v_y) location may be extracted, giving the observation vector x_v . The output of a node v is a location in a $D' \times D'$ spatial grid. This grid may be made relative to the location (v_x, v_y) , rendering the prediction task to be a relative displacement with respect to its initial position. This prediction is treated as a classification task and the model is trained with the cross entropy loss. In particular, the predictions from the RNN model may be taken, and a wrong prediction corrected if it deviates from the GT vertex by more than a threshold. The targets at training are then the relative displacements of each of these vertices with respect to GT.

[0086] Implementation details: In an embodiment, S is set to 1 and D' to 112. While the model may support much higher output resolutions, a larger D' may not justifiably improve results. The hidden state of the GRU in the GGNN has 256 dimensions. $T = 5$ propagation steps may be used. In the output model, f_1 is a 256×256 FC layer and f_2 is a $256 \times 15 \times 15$ MLP. In training, the predictions from the RNN are taken, and the vertices are replaced with GT vertices if they deviate by more than 3 cells.

[0087] 1.6. Annotating New Domains via Online Fine-Tuning

[0088] To simulate annotation of a completely new dataset, building off of a model trained on another, an online fine-tuning scheme may be used, which exploits a human-in-the-loop for annotation. Where C is the number of chunks the new data is divided into, CS is the chunk size, N_{EV} is the number of training steps for the evaluator and N_{MLE} , N_{RL} are the number of training steps for each chunk with MLE and RL, respectively. An example online fine-tuning is described in Algorithm 1 where PredictAndCorrect refers to the (simulated) annotator in the loop. Where training is on corrected data, the targets for MLE training may be smoothed with a manhattan distance transform truncated at distance 2.

Algorithm 1: Online Fine Tuning on New Datasets

```

bestPoly = cityscapesPoly;
while currChunk in (1..C) do
    rawData = readChunk(currChunk);
    data = PredictAndCorrect(rawData, bestPoly);
    data += SampleFromSeenData(CS);
    newPoly = TrainMLE(data, NMLE, bestPoly);
    newPoly = TrainRL(data, NRL, newPoly);
    newPoly = TrainEV(data, NEV, newPoly);
    bestPoly = newPoly;
end

```

[0089] 2. Experimental Results

[0090] Herein is presented an evaluation of embodiments of the model described above. Both automatic and interactive instances of annotation results on the Cityscapes dataset presented in D2 (M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. *The cityscapes dataset for semantic urban scene understanding*. In *CVPR*, 2016) are discussed and compared to strong pixel-wise methods. The generalization capability of the model is then characterized with evaluation on the KITTI dataset presented in D3 (A. Geiger, P. Lenz, and R. Urtasun. *Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite*. In *CVPR*, 2012) and four out-of-domain datasets spanning general scenes presented in D4 (B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. *Scene parsing through ade20k dataset*. In *CVPR*, 2017), aerial scenes presented in D5 (X. Sun, C. M. Christoudias, and P. Fua. *Free-shape polygonal object localization*. In *ECCV*, 2014), and medical imagery presented in D6 (A. H. Kadish, D. Bello, J. P. Finn, R. O. Bonow, A. Schaechter, H. Subacius, C. Albert, J. P. Daubert, C. G. Fonseca, and J. J. Goldberger. *Rationale and Design for the Defibrillators to Reduce Risk by Magnetic Resonance Imaging Evaluation (DETERMINE) Trial*. *J Cardiovasc Electrophysiol*, 20(9):982–7, 2009) and D7 (S. Gerhard, J. Funke, J. Martel, A. Cardona, and R. Fetter. *Segmented anisotropic ssTEM dataset of neural tissue*. *figshare*, 2013).

Finally, the online fine-tuning scheme is evaluated, demonstrating significant decrease in annotation time for novel datasets. It is assumed that user-provided ground-truth boxes around objects are given. Robustness of the model to noise is further analyzed with respect to those boxes, mimicking noisy annotators.

[0091] 2.1. In-Domain Annotation

[0092] The above model is first evaluated in both training and evaluating using the same domain. This mimics the scenario where one takes an existing dataset, and uses it to annotate novel images from the same domain. The Cityscapes dataset is currently one of the most comprehensive benchmarks for instance segmentation. It contains 2975 training, 500 validation and 1525 test images with 8 semantic classes. To ensure a fair comparison, the same alternative split is followed as proposed by D1. Ground-truth polygons may contain occluded parts of an instance, which are removed from the pixel-wise labelling using depth ordering. Following D1 the polygons are preprocessed according to depth ordering to obtain polygons for only visible regions of each instance.

[0093] Evaluation Metrics: Two quantitative measures are utilized to evaluate the model: 1) the intersection over union ('IoU') metric is used to evaluate the quality of the generated polygon and 2) the number of annotator clicks required to correct the predictions made by the model is counted. The correction protocol is described in detail below.

[0094] Baselines: Following D1, performance is compared with DeepMask disclosed in D8 (*P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. In NIPS, pages 1990–1998, 2015*), SharpMask disclosed in D9 (*P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. 2016*), as well as Polygon-RNN disclosed in D1 as state-of-the-art baselines. The first two approaches are pixel-wise methods and errors in their output may not be easily corrected by an annotator. The automatic mode of the model disclosed herein is compared. In their original approach, D8 and D9 exhaustively sample patches at different scales over the entire image. Here, we evaluate D8 and D9 by providing exact ground-truth boxes to their models.

[0095] As in D1, two additional baselines are also used, namely SquareBox disclosed in D1, and Dilation10 disclosed in D10 (*F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. ICLR, 2016*). SquareBox considers the provided bounding box as its prediction. Dilation10 is obtained from the segmentation results of D10 from the model that was trained on the Cityscapes dataset.

[0096] **Automatic Model:** The present model, Polygon-RNN++, is compared to the baselines in Table 1, and the results are presented as a series of submodels in which components are added to the base Polygon-RNN model, the addition of GGNN to the other components forming the full Polygon-RNN++. Here, Residual Polygon-RNN refers to the original Polygon-RNN model disclosed in D1 with the novel image architecture instead of VGG. The results of further aspects of the model added on are provided below the results for Residual Polygon-RNN. The full approach outperforms the top performer, Polygon-RNN as disclosed by D1, by almost 10% IoU, and achieves best performance for each class. Polygon-RNN++ also surpasses the reported human agreement in D1 of 78.6% IoU on cars, on average. Using human agreement on cars as a proxy, the model also obtains human-level performance for the truck and bus classes.

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	Mean
Square Box	35.41	53.44	26.36	39.34	54.75	39.47	46.04	26.09	40.11
Dilation10	46.80	48.35	49.37	44.18	35.71	26.97	61.49	38.21	43.89
DeepMask	47.19	69.82	47.93	62.20	63.15	47.47	61.64	52.20	56.45
SharpMask	52.08	73.02	53.63	64.06	65.49	51.92	65.17	56.32	60.21
Polygon-RNN	52.13	69.53	63.94	53.74	68.03	52.07	71.17	60.58	61.40
Residual Polygon-RNN	54.86	69.56	67.05	50.20	66.80	55.37	70.05	63.40	62.16
+ Attention	56.47	73.57	68.15	53.31	74.08	57.34	75.13	65.42	65.43
+ RL	57.38	75.99	68.45	59.65	76.31	58.26	75.68	65.65	67.17
+ Evaluator Network	62.34	79.63	70.80	62.82	77.92	61.69	78.01	68.46	70.21
+ GGNN	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	71.38

TABLE 1

[0097] **Interactive Mode:** The interactive mode aims to minimize annotation time while obtaining high quality annotations. Following the simulation proposed in D1, the number of annotator clicks required to correct predictions from the model is calculated. The annotator corrects a prediction if it deviates from the corresponding GT vertex by a min distance of T, where the hyperparameter T governs the quality of the produced annotations. For fair comparison, distances are computed using manhattan distance at the model output resolution using distance thresholds $T \in [1, 2, 3, 4]$, as in D1.

[0098] Additionally, a second threshold T_2 is introduced, which is defined as the IoU between the predicted polygon and the GT mask, where polygons achieving agreement above T_2 are considered unnecessary for the annotator to interfere. This threshold is exploited due to the somewhat unsatisfactory correction simulation above: for example, if the predicted vertex falls along a GT polygon edge, this vertex is in fact correct and should not be corrected. Note that, in the extreme case of $T_2 = 1$, the simulator assumes that corrections are necessary for every predicted polygon. In this case, the simulation is equivalent to the one presented in D1.

[0099] In FIG. 6, the average number of clicks per instance required to annotate all classes on the Cityscapes val set (500 images) with different values of T_2 is compared to Polygon-RNN of D1 at $T_2 = 1$. Using $T_2 = 1$, the present model outperforms the model of D1, requiring fewer clicks to obtain the same IoU. At $T_2 = 0.8$ the present model is still more accurate than Polygon-RNN, as disclosed in D1, at $T_2 = 1.0$. At $T_2 = 0.7$, over 80% mIoU is achieved with only 5 clicks per object on average, which is a reduction of more than 50% over D1. As indicated in Table 3, a hired human annotator takes about 96 clicks to achieve 78.6 mIoU while our model gets 88.3 mIoU with only 3.75 clicks. FIG. 7 shows frequency of required corrections for different T at $T_2 = 0.8$.

	GT	Human (crops)	PolyRNN*	Ours*	Ours
# Clicks	33.56	96.09	5.41	3.75	0
IoU (%)	100	78.6	85.73	88.31	80.19
Speed-Up	1x	-	6.20x	8.95x	∞

TABLE 3

[00100] **Robustness to bounding box noise:** To simulate the effect of a lazy annotator, the effect of noise in the bbox provided to the model is analyzed. The bbox is randomly expanded by a percentage of its width and height. Results in Table 5 illustrates that the present model is very robust to some amount of noise (0-5%). Even in the presence of moderate and extreme noise (5-10%,10-15%), it outperforms the reported performance of previous baselines which use perfect bboxes.

Bbox Noise (%)	IoU (%)
0	71.38
0-5	70.54
5-10	68.07
10-15	64.80

TABLE 5

[00101] 2.2. Cross-Domain Evaluation

[00102] In this section, the performance of the present model is evaluated on different datasets that capture both shifts in environment (KITTI, as disclosed in D3) and domain (general scenes, aerial, medical). The model used was trained on Cityscapes without any fine-tuning on these datasets.

[00103] **KITTI:** Polygon-RNN++ is used to annotate 741 instances of KITTI. The results in automatic mode are reported in TABLE 4 and the performance with a human in the loop is illustrated in FIG. 8. The present method outperforms all baselines showcasing its robustness to change in environment while being in a similar domain. With an annotator in the loop, the present model requires on average 5 fewer clicks than D1 to achieve the same IoU. It achieves human level agreement of 85% as reported by D11 (*L.-C. Chen, S. Fidler, A. Yuille, and R. Urtasun. Beat the mturkers: Automatic image labeling from weak 3d supervision. In CVPR, 2014*) with only 2 clicks on average by the annotator.

Model	# of Clicks	IoU (%)
DeepMask	-	78.3
SharpMask	-	78.8
Beat The MTurkers	0	73.9
Polygon-RNN	0	74.22
Ours w/o GGNN	0	81.40
Ours w/ GGNN	0	83.14

TABLE 4

[00104] 2.2.1 Out-of-Domain Imagery

[00105] Datasets exhibiting varying levels of domain shift from Cityscapes are considered to evaluate the generalization capabilities of the present model.

[00106] ADE20K: The ADE20K disclosed in D4 is a general scene parsing dataset containing 20,210 images in the training set, 2,000 images in the validation set, and 3,000 images in the testing set. The following subset of categories are selected from the validation set: television receiver, bus, car, oven, person and bicycle in our evaluation.

[00107] Aerial Imagery: The Aerial Rooftop dataset disclosed in D5 consists of 65 aerial images of rural scenes containing several building rooftops, a majority of which exhibit fairly complex polygonal geometry. Performance for this dataset is reported for the test set.

[00108] Medical Imagery [14, 31, 10]: Two medical segmentation datasets are used; one disclosed in D6 and D12 (*A. Suinesiaputra, B. R. Cowan, A. O. Al-Agamy, M. A. Elattar, N. Ayache, A. S. Fahmy, A. M. Khalifa, P. Medrano-Gracia, M.-P. Jolly, A. H. Kadish, D. C. Lee, J. Margeta, S. K. Warfield, and A. A. Young. A collaborative resource to build consensus for automated left ventricular segmentation of cardiac MR images. Medical Image Analysis, 18(1):50 – 62, 2014*) and the other disclosed in D7 for our experiments. The former, used in the

Left Ventricle Segmentation Challenge disclosed in D12, divides the data of 200 patients equally in the training and validation sets. The performance of the present model is reported on a subset of the validation set which only includes the outer contours that segment the epicardium. The latter provides two image stacks (training and testing) each containing 20 sections from serial section Transmission Electron Microscopy (ssTEM) images of the ventral nerve cord. The mitochondria and synapse segmentations are used from this data for the present model. Since ground-truth instances for the test stack are not publicly available, evaluation is done on the training set.

[00109] Quantitative Results: For out-of-domain datasets, a baseline named Ellipse is introduced, which fits an ellipse into the GT bounding box. This was used in the present tests as many of the instances in D12 were ellipses. Results are shown with perfect and expanded bounding boxes (expansion similar to the present model) for Square Box and Ellipse. DeepMask and SharpMask were evaluated with perfect bounding boxes with the threshold suggested by the authors. Table 2, depicting some of the results, demonstrates high generalization capabilities of the present model.

Model	ADE	Rooftop	Cardiac MR	ssTEM
SquareBox (Expansion)	42.95	40.71	62.10	42.24
Ellipse (Expansion)	48.53	47.51	73.63	51.04
Square Box (Perfect)	69.35	62.11	79.11	66.53
Ellipse (Perfect)	69.53	66.82	92.44	71.32
DeepMask	59.74	15.82	60.70	31.21
SharpMask	61.66	18.53	69.33	46.67
Ours w/o GGNN	70.21	65.03	80.55	53.77
Ours w/ GGNN	71.82	65.67	80.63	53.12

TABLE 2

[00110] Online Fine-tuning: In these experiments, the simulated annotator has parameters $T = 1$ and $T_2 = 0.8$. FIG. 9 reports the percentage of clicks saved with respect to GT polygons for the Cityscapes model and the online fine-tuned models. The adaptive approach overcomes stark domain shifts with as few as one chunk of data (40 images for Sunnybrook, 3 for ssTEM, 200 for

ADE and 20 for Aerial) showcasing strong generalization. Overall, at least 65% overall reduction in the number of clicks across all datasets, with the numbers almost at 100% for the Sunnybrook Cardiac MR dataset. These results indicated that an annotation tool may be able to learn along with the annotator and significantly reduce human effort.

[00111] 2.3. Qualitative Results

[00112] FIG. 10 shows example predictions obtained in automatic mode on Cityscapes. The improvements from specific parts of the model are illustrated in FIG. 11. As indicated, using RL and the evaluator may lead to crisper predictions, while the GGNN may upscale, add points and build a polygon resembling human annotation. FIG. 12 showcases automatic predictions from PolygonRNN++ on the out-of-domain datasets. The labeling results shown in FIG. 12 are obtained by exploiting GT bounding boxes, and no fine-tuning.

[00113] FIG. 13 illustrates a visualization of attention maps in the present model with T set to various levels.

[00114] 3. Method Embodiment

[00115] An embodiment of a method 14000 is depicted in FIG. 14. Method 14000 is a method of annotating an object, by representing the object as a polygon outline. Method 14000 includes the step 14001 of receiving an image depicting an object. An object so received may then be processed by a CNN encoder to generate one or more image features at step 14002. One or more image features are then used to generate one or more first vertex predictions at step 14003, where the first vertex predictions are used by a subsequent neural network as a basis for generating a polygon representation of an image object.

[00116] One or more polygon representations are then generated at step 14004 by a recurrent decoder. The recurrent decoder may take the one or more first vertex predictions and use each first vertex prediction to create a polygon representation, which representations together may form a set of possible polygon representations. At step 14005 an evaluator network may select an object annotation selection from among the set of possible polygon representations.

[00117] The selected object annotation is then fed into a gated graph neural network provided to increase the resolution of the polygon representation at step 14006. The GGNN may produce a higher resolution polygon by adding a set of supplementary vertex predictions to the set of primary vertex predictions which defines the selected object annotation, defining a propagation model, and applying the model to adjust the position of the vertices of the set of supplementary vertex predictions and the vertices of the set of primary vertex predictions. The gated graph neural network may use input from the CNN encoder and the Recurrent decoder.

[00118] According to an embodiment, the number of vertices is decided by the model itself as it generates an end of sequence token when it thinks the polygon is completed.

[00119] The resulting higher resolution polygon object annotation may then be applied, such as to an automated system, such as a system for automated driving, map annotation, or medical image annotation.

[00120] 4. System Embodiment

[00121] In some embodiments, as depicted in FIG. 15, a system 15000 is provided to annotate one or more objects, such as to locate an object in an image. System 15000 includes an input unit 15010 for receiving an image or other object carrying item for annotation. System 15000 includes a system processor 15020 for processing input. System 15000 also includes an output or application unit 15030, such as a computer monitor or other display or an output unit for sending annotated objects to a system such as an automated driving system, a medical imaging system, or a mapping system.

[00122] In some embodiments, object annotation is applied to guide an automated system. For example, object annotation is applied to guide the operation of an autonomous driving system or to guide automated aspects or features of a driving system.

[00123] As depicted in FIG. 15, system processor 15020 may include a CNN Encoder 15021, for generating image features from a received image. Image features may then be used to predict a first vertex of an object annotation, to be used by recurrent decoder 15022. Recurrent decoder 15022 may include an attention unit 15023 for visual attention, and an evaluator network 15024

to select a preferred polygon representation from polygon representations produced by recurrent decoder 15022.

[00124] System processor 15020 may also include a gated graph neural network 15025 for producing an upscaled polygon representation of an object. GGNN 15025 may include a propagation block 15026 and an output block 15027. GGNN 15025 may receive output from CNN encoder 15021, such as edge information, and may receive information from recurrent decoder 15022, such as vertex information defining a polygon, such as a preferred polygon selected by evaluator network 15024.

[00125] While various elements, blocks, or units are depicted or described as being either independent or as being components of other elements, blocks, or units, in other embodiments other arrangements of elements, blocks or units may be employed.

[00126] 5. Further Method Embodiment

[00127] As depicted in FIG. 16, another embodiment of the present invention may be a method 16000 of training a system of object annotation. Method 16000 includes receiving a training dataset at step 16001, initiating a training sequence for setting one or more weight matrices of the object annotation system using managed learning environment training at step 16002, and fine-tuning the one or more weight matrices of the object annotation system using reinforcement learning to produce a trained object annotation system at step 16003.

[00128] Method 16000 may also include producing an object annotation prediction for an image of the training dataset at step 16004, submitting the object annotation prediction for human correction at step 16005, and producing a human correction of the object annotation prediction and feeding the human correction back into the object annotation system to further train the object annotation system at step 16006.

[00129] Method 16000 may also include including fine-tuning the object annotation system using online fine-tuning at step 16007. Online fine-tuning involves training the prediction system while a user interacts with the annotation tool or platform. As a user interacts and creates new labelled data, the model can be trained on it to produce better annotations in subsequent usage, leading to lesser human interaction in the future.

[00130] It is to be understood that while method steps have been described in a particular order and depicted as following one another sequentially, one or more method steps may be performed simultaneously, and in some embodiments method steps may be performed in orders other than described and depicted.

[00131] All neural networks, such as neural networks of system 15000, including a CNN of CNN encoder 15021 and an RNN of recurrent decoder 15022, may be implemented by one or more computers executing computer readable instructions found on computer readable medium.

[00132] Alternative Embodiment

[00133] As described above, vertices are predicted sequentially, however, additional speed may be gained via an implementation that predicts all vertices simultaneously.

[00134] In the alternative embodiment, object annotation is framed as a regression problem, where the locations of all vertices are predicted simultaneously. The object may be represented as a graph with a fixed topology, and perform prediction using a Graph Neural Network (GNN) such as a Graph Convolutional Network (GCN). The model may be used and optimized for interactive annotation. The framework may further allow for parametrization of objects with either polygons or splines, adding additional flexibility and efficiency to the interactive annotation process. This embodiment is referred to herein as Curve-GCN, and is end-to-end differentiable, and runs in real time.

[00135] Object Annotation via Curve-GCN

[00136] The framework for Curve-GCN annotates object instances with either polygons or (closed) splines. In order to approximate a curved contour, one would need to draw a polygon with a significant number of vertices, while this could be efficiently handled with a few control points using splines. The framework is designed to enable both a polygon and a spline representation of an object contour.

[00137] The typical labeling scenario is followed where it is assumed that the annotator has selected the object of interest by placing a bounding box around it (see Acuna and Castreion). The image is cropped around this box and frame object annotation inside this crop as a regression

problem; to predict the locations of all control points (vertices) simultaneously, from an initialization with a fixed topology. The model is described from representation to inference first, then a discussion of training and finally an analysis of using the model for human-in-the loop annotation, by formulating both inference as well as training in the interactive regime.

[00138] Polygon/Spline-GCN

[00139] Assume the target object shapes can be well represented using N control points, which are connected to form a cycle. The induced shape is rendered by either connecting them with straight lines (thus forming a polygon), or higher order curves (forming a spline). Treat the location of each control point as a continuous random variable, and learn to predict these via a Graph Neural Network that takes image evidence as input. In Acuna, the authors exploited Gated Graph Neural Networks (GGNN) [Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel. Gated graph sequence neural networks, *ICLR*, 2016.] as a polygon refinement step, in order to upscale the vertices output by the RNN to a higher resolution. In similar vein, Pixel2Mesh [N. Wang, Y. Zhang, Z. Li, Y. Fu, W. Liu, and Y.-G. Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images, *ECCV*, 2018] exploited a Graph Convolutional Network (GCN) to predict vertex locations of a 3D mesh. The key difference between a GGNN and a GCN is in the graph information propagation; a GGNN shares propagation matrices through time akin to a gated recurrent unit (GRU), whereas a GCN has propagation steps implemented as unshared “layers”, similar to a typical CNN architecture. The GCN is adopted in the present model due to its higher capacity. Hence, the reference name of the present model, Curve-GCN, which includes Polygon or Spline-GCN.

[00140] **Notation:** Initialize the nodes of the GCN to be at a static initial central position in the given image crop (FIG. 17). The GCN predicts a location offset for each node, aiming to move the node correctly onto the object’s boundary. Let $\mathbf{cp}_i = [x_i, y_i]^T$ denote the location of the i -th control point and $V = \{\mathbf{cp}_0, \mathbf{cp}_1, \dots, \mathbf{cp}_{N-1}\}$ be the set of all control points. Define the graph to be $G = (V, E)$, with V defining the nodes and E the edges in the graph. Form E by connecting each vertex in V with its four neighboring vertices. This graph structure defines how the information propagates in the GCN. Connecting 4-way allows faster exchange of information between the nodes in the graph.

[00141] Extracting Features: Given a bounding box, crop the corresponding area of the image and encode it using a CNN, the specific choice of which is determined by the user. Denote the feature map obtained from the last convolutional layer of the CNN encoder applied on the image crop as F_c . In order to help the model see image boundaries, supervise two additional branches, i.e. an edge branch and a vertex branch, on top of the CNN encoder's feature map F_c , both of which consist of one 3×3 convolutional layer and one fully-connected layer. These branches are trained to predict the probability of existence of an object edge/vertex on a 28×28 grid. Train these two branches with the binary cross entropy loss. The predicted edge and vertices outputs are concatenated with F_c , to create an augmented feature map F . The input feature for a node $\mathbf{cp}_i$ in the GCN is a concatenation of the node's current coordinates (x_i, y_i) , where top-left of the cropped images is $(0, 0)$ and image length is 1, and features extracted from the corresponding location in F : $f_i^0 = \text{concat}\{F(x_i, y_i), x_i, y_i\}$. Here, $F(x_i, y_i)$ is computed using bilinear interpolation.

[00142] GCN Model: A multi-layer GCN is used. The graph propagation step for a node $\mathbf{cp}_i$ at layer l is expressed as:

$$f_i^{l+1} = w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l$$

[00143] where $\mathcal{N}(\mathbf{cp}_i)$ denotes the nodes that are connected to $\mathbf{cp}_i$ in the graph, and w_0^l, w_1^l are the weight matrices. Following the method of Bronstein [M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data, *CVPR*, 2017], utilize a Graph-ResNet to propagate information between the nodes in the graph as a residual function. The propagation step in one full iteration at layer l then takes the following form:

$$\begin{aligned} r_i^l &= \text{ReLU}(w_0^l f_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} w_1^l f_j^l) \\ r_i^{l+1} &= \tilde{w}_0^l r_i^l + \sum_{\mathbf{cp}_j \in \mathcal{N}(\mathbf{cp}_i)} \tilde{w}_1^l r_j^l \\ f_i^{l+1} &= \text{ReLU}(r_i^{l+1} + f_i^l), \end{aligned}$$

[00144] where $\tilde{w}_0, \tilde{w}_1$ are weight matrices for the residual. On top of the last GCN layer, apply a single fully connected layer to take the output feature and predict a relative location shift, $(\Delta x_i,$

Δy_i), for each node, placing it into location $[x'_i, y'_i] = [x_i + \Delta x_i, y_i + \Delta y_i]$. Also perform iterative inference similar to the coarse-to-fine prediction in [35]. To be more specific, the new node locations $[x_i, y_i]$ are used to re-extract features for the nodes, and another GCN predicts a new set of offsets using these features. This mimics the process of the initial polygon/spline iteratively “walking” towards the object’s boundaries.

[00145] Spline Parametrization: The choice of spline is important, particularly for the annotator’s experience. The two most common splines, i.e. the cubic Bezier spline and the uniform B-Spline [e.g. H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-spline techniques*. Springer Science & Business Media, 2013], are defined by control points which do not lie on the actual curve, which could potentially confuse an annotator that needs to make edits. Following Tan [J. H. Tan and U. R. Acharya. Active spline model: a shape based model interactive segmentation. *Digital Signal Processing*, 35:64–74, 2014], use the centripetal Catmull-Rom spline (CRS) [e.g. C. Yuksel, S. Schaefer, and J. Keyser. Parameterization and applications of catmull-rom curves. *Computer-Aided Design*, 43(7):747–755, 2011.], which has control points along the curve. Yuksel et al., which is incorporated herein by reference, provides for a detailed visualization of different types of splines.

[00146] For a curve segment S_i defined by control points $\mathbf{cp}_{i-1}$, $\mathbf{cp}_i$, $\mathbf{cp}_{i+1}$, $\mathbf{cp}_{i+2}$ and a knot sequence t_{i-1} , t_i , t_{i+1} , t_{i+2} , the CRS is interpolated by:

$$S_i = \frac{t_{i+1}-t}{t_{i+1}-t_i} L_{012} + \frac{t-t_i}{t_{i+1}-t_i} L_{123}$$

[00147] where

$$\begin{aligned} L_{012} &= \frac{t_{i+1}-t}{t_{i+1}-t_{i-1}} L_{01} + \frac{t-t_{i-1}}{t_{i+1}-t_{i-1}} L_{12} \\ L_{123} &= \frac{t_{i+2}-t}{t_{i+2}-t_i} L_{12} + \frac{t-t_i}{t_{i+2}-t_i} L_{23} \\ L_{01} &= \frac{t_i-t}{t_i-t_{i-1}} \mathbf{cp}_{i-1} + \frac{t-t_{i-1}}{t_i-t_{i-1}} \mathbf{cp}_i \\ L_{12} &= \frac{t_{i+1}-t}{t_{i+1}-t_i} \mathbf{cp}_i + \frac{t-t_i}{t_{i+1}-t_i} \mathbf{cp}_{i+1} \\ L_{23} &= \frac{t_{i+2}-t}{t_{i+2}-t_{i+1}} \mathbf{cp}_{i+1} + \frac{t-t_{i+1}}{t_{i+2}-t_{i+1}} \mathbf{cp}_{i+2}, \end{aligned}$$

[00148] and $t_{i+1} = \|\mathbf{cp}_{i+1} - \mathbf{cp}_i\|_2^\alpha + t_i$, $t_0 = 0$. Here, α ranges from 0 to 1. A choice is made of $\alpha = 0.5$ following Tan, which in theory produces splines without cusps or self-intersections

[see Yuskel et al., *infra*]. To make the spline a closed and C^1 -continuous curve, add three additional control points:

$$\begin{aligned} \mathbf{cp}_N &= \mathbf{cp}_0 \\ \mathbf{cp}_{N+1} &= \mathbf{cp}_0 + \frac{\|\mathbf{cp}_{N-1} - \mathbf{cp}_0\|_2}{\|\mathbf{cp}_1 - \mathbf{cp}_0\|_2} (\mathbf{cp}_1 - \mathbf{cp}_0) \\ \mathbf{cp}_{-1} &= \mathbf{cp}_0 + \frac{\|\mathbf{cp}_1 - \mathbf{cp}_0\|_2}{\|\mathbf{cp}_{N-1} - \mathbf{cp}_0\|_2} (\mathbf{cp}_{N-1} - \mathbf{cp}_0) \end{aligned}$$

[00149] Training

[00150] The model is trained with two different loss functions. First, the model is trained with a Point Matching Loss, and then fine-tuned with a Differentiable Accuracy Loss. More specific details and ablations are provided in the experimental data.

[00151] Point Matching Loss

[00152] Typical point-set matching losses, such as the Chamfer Loss, assumed unordered sets of points (i.e. they are permutation invariant). A polygon/spline, however, has a well-defined ordering, which an ideal point set matching loss would obey. Assuming equal sized and similarly ordered (clockwise or counter-clockwise) prediction and ground truth point sets, denoted as $\mathbf{p} = \{p_0, p_1, \dots, p_{K-1}\}$, and $\mathbf{p}' = \{p'_0, p'_1, \dots, p'_{K-1}\}$ respectively (K is the number of points), define the matching loss as:

$$L_{\text{match}}(\mathbf{p}, \mathbf{p}') = \min_{j \in [0 \dots, K-1]} \sum_{i=0}^{K-1} \|p_i - p'_{(j+i)\%K}\|_1$$

[00153] Notice that this loss explicitly ensures an order in the vertices in the loss computation. Training with an unordered point set loss function, while maintaining the topology of the polygon could result in catastrophic self-intersections, while the ordered loss function avoids it.

[00154] Sampling equal sized point sets. Since annotations may vary in the number of vertices, while the model always assumes N , additional points are sampled along boundaries of both ground-truth polygons and our predictions. For Polygon-GCN, K points are uniformly sampled along edges of the predicted polygons, and for Spline-GCN, K points are sampled along the spline by

uniformly ranging t from t_i to t_{i+1} . The same number of points are uniformly sampled along the edges of the ground-truth polygon. $K = 1280$ is used in the experiments as detailed below. Sampling more points would have a higher computational cost, while sampling fewer points would make curve approximation less accurate. Note that the sampling only involves interpolating the control points, ensuring differentiability.

[00155] Differentiable Accuracy Loss

[00156] To perfectly align the predicted polygon and the ground-truth silhouette, a differentiable rendering loss is employed, which encourages masks rendered from the predicted control points to agree with ground-truth masks. This has been used previously to optimize 3D mesh vertices to render correctly onto a 2D image [e.g. H. Kato, Y. Ushiku, and T. Harada. Neural 3d mesh renderer, *ECCV*, 2018, and M. M. Loper and M. J. Black. Opendr: An approximate differentiable renderer, *ECCV*, pages 154–169, 2014, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors].

[00157] The rendering process can be described as a function $R; M(\theta) = R(\mathbf{p}(\theta))$, where $\mathbf{p}$ is the sampled point sequence on the curve, and M is the corresponding mask rendered from $\mathbf{p}$. The predicted and the ground-truth masks can be compared by computing their difference with the $L1$ loss:

$$L_{\text{render}}(\theta) = \|M(\theta) - M_{\text{gt}}\|_1$$

[00158] Note that L_{render} is exactly the pixel-wise accuracy of the predicted mask $M(\theta)$ with respect to the ground truth M_{gt} . The method for obtaining M in the forward pass and back-

propagating the gradients through the rendering process R , from $\frac{\partial L}{\partial M}$ to $\frac{\partial L}{\partial \mathbf{p}}$ in the backward pass are detailed next.

[00159] **Forward Pass:** Render $\mathbf{p}$ into a mask using OpenGL. As shown in FIG. 18 the shape is decomposed into triangle fans $\mathbf{f}_j$ and assign positive or negative values to their area based on their orientation. Render each face with the assigned value, and sum over the rendering of all the triangles to get the final mask. Note that this works for both convex and concave polygons [e.g. D. Shreiner and T. K. O. A. W. Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1*. Addison-Wesley Professional, 7th edition, 2009].

[00160] **Backward Pass:** The rendering process is non-differentiable in OpenGL due to rasterization, which truncates all float values to integers. However, following Loper et al., *infra*, compute its gradient with first order Taylor expansion. Then reutilize the triangle fans from the decomposition in the forward pass (see FIG. 18) and analyze each triangle fan separately. Taking a small shift of the fan $\mathbf{f}_j$, calculate the gradient with respect to the j -th triangle as:

$$\frac{\partial M_j}{\partial \mathbf{f}_j} = \frac{R(\mathbf{f}_j + \Delta t) - R(\mathbf{f}_j)}{\Delta t},$$

[00161] where M_j is the mask corresponding to the fan $\mathbf{f}_j$. Here, Δt can be either in the x or y direction. For simplicity, let Δt to be a 1 pixel shift, which alleviates the need to render twice, and allows calculating gradients by subtracting neighboring pixels. Next, pass gradient: $\frac{\partial M_j}{\partial \mathbf{f}_j}$ to its three vertices $\mathbf{f}_{j,0}$, $\mathbf{f}_{j,1}$ and $\mathbf{f}_{j,2}$:

$$\frac{\partial M_j}{\partial \mathbf{f}_{j,k}} = \sum_i w_k^i \frac{\partial M_j^i}{\partial \mathbf{f}_j} \quad k \in [0, 1, 2]$$

which is summed over all pixels i . For the i -th pixel M_j^i in the rendered image M_j , compute its weight w_0^i , w_1^i and w_2^i with respect to the vertices of the face $\mathbf{f}_j$ as its barycentric coordinates. For more details, refer to Loper et al.

[00162] **Annotator in The Loop**

[00163] A potential drawback of Polygon-RNN is that once the annotator corrects one point, all of the subsequent points will be affected due to the model's recurrent structure. This is often undesirable, as the changes can be drastic. Alternatively, it is desired to have flexibility to change any point, and further constrain that only the neighboring points can change. As in Polygon-RNN, the correction is assumed to be in the form of drag-and-drop of a point.

[00164] To make the model interactive, another GCN is trained that consumes the annotator's correction and predicts the relative shifts of the other control points. This GCN is referred to herein as the InteractiveGCN. The network's architecture the same as the original GCN, except that two

additional dimensions are now appended to the corrected node's (say node i) input feature, representing the annotator's correction:

$$f_i^0 = \text{concat}\{F(x_i, y_i), x_i, y_i, \Delta x_i, \Delta y_i\},$$

[00165] where $(\Delta x_i, \Delta y_i)$ is the shift given by the annotator. For all other nodes, set $(\Delta x_i, \Delta y_i)$ to zero. Do not perform iterative inference here. The InteractiveGCN allows a radius of influence by simply masking predictions of nodes outside the radius to 0. In particular, let k neighbors on either side of node i to be predicted, *i.e.*, $\mathbf{cp}_{(i-k)\%N}, \dots, \mathbf{cp}_{(i-1)\%N}, \mathbf{cp}_{(i+1)\%N}, \dots, \mathbf{cp}_{(i+k)\%N}$. Set $k = 2$ is set in the experiments described herein, while noting that in principle, the annotator could vary k at test time.

[00166] InteractiveGCN is trained by mimicking an annotator that iteratively moves wrong control points onto their correct locations. This assumes that the annotator always chooses to correct the worst predicted point. This is computed by first aligning the predicted polygon with GT, by finding the mini-mum of our point matching loss (Sec. 3.2.1). Then find the point with the largest maximum manhattan distance to the corresponding GT point. The network is trained to move the neighboring points to their corresponding ground-truth positions. Then iterate between the annotator choosing the worst prediction, and training to correct its neighbors. In every iteration, the GCN first predicts the correction for the neighbors based on the last annotator's correction, and then the annotator corrects the next worst point. Then let the gradient back-propagate through the iterative procedure, helping the InteractiveGCN to learn to incorporate possibly many user interactions. The training procedure is summarized in Algorithm 1, where c denotes the number of iterations.

Algorithm 1 Learning to Incorporate Human-in-the-Loop

```

1: while not converged do
2:   (rawImage, gtCurve) = Sample(Dataset)
3:   (predCurve,  $F$ ) = Predict(rawImage)
4:   data = []
5:   for i in range(c) do
6:     corrPoint = Annotator(predictedCurve)
7:     data += (predCurve, corrPoint, gtCurve,  $F$ )
8:     predCurve = InteractiveGCN(predCurve, corrPoint)
9:                                     ▷ Do not stop gradients
10:  TrainInteractiveGCN(data)

```

[00167] Experimental Results

[00168] Curve-GCN was tested for both in-domain and cross-domain instance annotation. The Cityscapes dataset from Cordts et al. [M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding, *CVPR*, 2016] was used as the main benchmark to train and test for the Curve-GCN model. Both automatic and interactive regimes and analyzed, and compared to state-of-the-art baselines for both. For cross-domain experiments, the Cityscapes-trained model is evaluated for generalization capability on the KITTI dataset from Geiger et al. [A. Geiger, P. Lenz, and R. Urtasun. Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite, *CVPR*, 2012] and four out-of-domain datasets, ADE20K from Zhou et al. [B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ade20k dataset, *CVPR*, 2017], Aerial Rooftop from Sun et al. [X. Sun, C. M. Christoudias, and P. Fua. Free-shape polygonal object localization. In *European Conference on Computer Vision*, pages 317–332. Springer, 2014], Cardiac MR from Suinesiaputra et al. [A. Suinesiaputra, B. R. Cowan, A. O. Al-Agamy, M. A. Elat-tar, N. Ayache, A. S. Fahmy, A. M. Khalifa, P. Medrano-Gracia, M.-P. Jolly, A. H. Kadish, et al. A collaborative resource to build consensus for automated left ventricular segmentation of cardiac mr images. *Medical image analysis*, 18(1):50–62, 2014], and ssTEM from Gerhard et al. [S. Gerhard, J. Funke, J. Martel, A. Cardona, and R. Fetter. Segmented anisotropic ssTEM dataset of neural tissue. 11 2013], following those used for Polygon-RNN++ as previously described. To

For the purpose of object annotation, boundaries are especially important – even slight deviations may not escape the eye of an annotator. Thus, the Boundary F score is also computed per Perazzi et al. [Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung. A benchmark dataset and evaluation methodology for video object segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016], which calculates precision/recall between the predicted and the ground-truth boundary, by allowing some misalignment slack. Given that Cityscapes is finely annotated, results are reported at stringent thresholds (slack) of 1 and 2 pixels.

[00174] In-Domain Annotation

[00175] The model is first valuated when both training and inference are performed on Cityscapes. This dataset contains 2975/500/1525 images for training, validation and test, respectively. For a fair comparison, the same split and data preprocessing procedure is followed as in Polygon-RNN++ above.

[00176] **Automatic Mode:** Table 6 reports results of the Polygon and Spline-GCN and compares them with baselines. The performance metric used is IoU. Note that PSP-DeepLab uses a more powerful image encoder, which is pretrained on PASCAL for segmentation. The Spline-GCN outperforms Polygon-RNN++ and is on par with PSP-DeepLab. It also performs over Polygon-GCN, likely because most Cityscapes objects are curved. The results also show the significance of the differentiable accuracy loss (diffAcc) which leads to large improvements over the model trained with the matching loss alone (denoted with MLoss in Table). The model mostly loses against PSP-DeepLab on the train category, which is believed due to the fact that trains in Cityscapes are often occluded and broken into multiple components. Since the GCN approach predicts only a single connected component, it may struggle on such cases.

Model	Bicycle	Bus	Person	Train	Truck	Motorcycle	Car	Rider	Mean
Polygon-RNN++	57.38	75.99	68.45	59.65	76.31	58.26	75.68	65.65	67.17
Polygon-RNN++ (with BS)	63.06	81.38	72.41	64.28	78.90	62.01	79.08	69.95	71.38
PSP-DeepLab	67.18	83.81	72.62	68.76	80.48	65.94	80.45	70.00	73.66
Polygon-GCN (MLoss)	63.68	81.42	72.25	61.45	79.88	60.86	79.84	70.17	71.19
+ DiffAcc	66.55	85.01	72.94	60.99	79.78	63.87	81.09	71.00	72.66
Spline-GCN (MLoss)	64.75	81.71	72.53	65.87	79.14	62.00	80.16	70.57	72.09
+ DiffAcc	67.36	85.43	73.72	64.40	80.22	64.86	81.88	71.73	73.70

Table 6

[00177] Table 2 compares models with respect to F boundary metrics. It may be observed that while Spline-GCN is on par with PSP-DeepLab under the IoU metric, it is significantly better in the more precise F score. This means that the model more accurately aligns with the object boundaries than PSP-DeepLab. Qualitative results are shown in Fig 20, 21, and 22.

Model	mIOU	F at 1px	F at 2px
Polymn++ (BS)	71.38	46.57	62.26
PSP-DeepLab	73.66	47.10	62.82
Spline-GCN	73.70	47.72	63.64
DEXTR	79.40	55.38	69.84
Spline-GCN-EXTR	79.88	57.56	71.89

Table 7

[00178] **Ablation Study:** Each component of the model is studied and provided results documented for both Polygon and Spline-GCN in Table 8. Performing iterative inference leads to a significant boost, and adding the boundary branch to the CNN further improves performance.

Model	Spline	Polygon
GCN	68.55	67.79
+ Iterative Inference	70.00	70.78
+ Boundary Pred.	72.09	71.19
+ DiffAcc	73.70	72.66

Table 8

[00179] **Additional Human Input:** In DEXTR per Maninis et al. [K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018], the authors proposed to use 4 extreme points on the object boundary as an effective information provided by the annotator. Compared to just a box, extreme points require 2 additional clicks. The GCN model is compared to DEXTR in this regime, and follows their strategy in how this information is provided to the model. To be specific, points (in the form of a heat map) are stacked with the image, and passed to a CNN. To compare with DEXTR, DeepLab-v2 is used, as per Maninis et al. The models are referred with such input by appending EXTR.

[00180] Note that the image crops used in Polygon-RNN, are obtained by extracting an image inside a square box (and not the actual box provided by the annotator). However, due to significant occlusion in Cityscapes, doing so leads to ambiguities, since multiple objects can easily fall in the

same box. By providing 4 extreme points, the annotator more accurately points to the target object. To verify how much accuracy is really due to the additional two clicks, an instantiation of the model is tested to which the four corners of the bounding box are provided as input. This is still a 2-click (box) interaction from the user, however, it reduces the ambiguity about which object to annotate. This model is referred to herein by appending BOX.

[00181] Since DEXTR labels pixels and thus more easily deals with multiple component instances, another instantiation of the model is proposed which still exploits 4 clicks on average, yet collects these differently. Specifically, the annotator is requested to provide a box around each component, rather than just a single box around the full object. On average, this leads to 2.4 clicks per object. This model is referred to with MBOX. To match the 4-click budget, the annotator clicks on the worst predicted boundary point for each component, which leads to 3.6 clicks per object, on average.

[00182] Table 9 shows that in the extreme point regime, the model is already better than DEXTR, whereas the alternative strategy is even better, yielding an 0.8% improvement overall with fewer clicks in average. The method also significantly outperforms DEXTR in the boundary metrics (FIG. 26).

Model	Bicycle	Bus	Person	Train	Truck	Mcycle	Car	Rider	Mean	# clicks
Spline-GCN-BOX	69.53	84.40	76.33	69.05	85.08	68.75	83.80	73.38	76.29	2
PSP-DEXTR	74.42	87.30	79.30	73.51	85.42	73.69	85.57	76.24	79.40	4
Spline-GCN-EXTR	75.09	87.40	79.88	72.78	86.76	73.93	86.13	77.12	79.88	4
Spline-GCN-MBOX	70.45	88.02	75.87	76.35	82.73	70.76	83.32	73.49	77.62	2.4
+ One click	73.28	89.18	78.45	79.89	85.02	74.33	85.15	76.22	80.19	3.6

Table 9

[00183] **Interactive Mode:** For interactive mode, an annotator is simulated correcting vertices, following the protocol discussed above for Polygon RNN++. In particular, the annotator iteratively makes corrections until the IoU is greater than a threshold T, or the model stops improving its prediction. The predicted curve achieving agreement above T is considered as a satisfactory annotation.

[00184] FIGs. 23A, 23B and 24 show IoU vs number of clicks at different thresholds T. The results are compared to Polygon-RNN++. The results show significant improvements over the

baseline, highlighting the model as a more efficient annotation tool. Performance is further analyzed when using 40 vs 20 control points. The version with fewer control points is slightly worse in automatic mode, however, it is almost on par in the interactive mode. This may suggest that coarse-to-fine interactive correction may be the optimal approach.

[00185] Inference Times: Timings are reported in Table 10. The model is an order of magnitude faster than Polygon-RNN++, running at 28.4 ms, while Polygon-RNN++ requires 298.0 ms. In the interactive mode, the model reuses the computed image features computed in the forward pass, and thus only requires 2.6ms to incorporate each correction. On the other hand, Polygon-RNN requires to run an RNN after every correction, thus still requiring 270 ms.

Model	Time(ms)
Polygon-RNN++	298.0
Polygon-RNN++ (Corr.)	270.0
Polygon-GCN	28.7
Spline-GCN	29.3
Polygon-GCN (Corr.)	2.0
Spline-GCN (Corr.)	2.6

Table 10

[00186] Cross-Domain Evaluation

[00187] The model is evaluated on its ability to generalize to new datasets. Generalization is crucial, in order to effectively annotate a variety of different imagery types. It may be shown that by fine-tuning on only a small set of the new dataset (10%) leads to fast adaptation to new domains.

[00188] Following Polygon-RNN++ and using the Cityscapes-trained model and test it on KITTI (in-domain driving dataset), ADE20k (general scenes), Rooftop (aerial imagery), and two medical datasets as previously described.

[00189] Quantitative Results. Table 11 provides the results. Simple baselines are adopted from Polygon-RNN++. The models are further fine-tuned (with dif-fAcc) with 10% randomly sampled training data from the new domain. Note that ssTEM does not have a training split, and thus is omitted for this dataset. Results show that the model generalizes better than PSP-DeepLab, and that fine-tuning on very little annotated data effectively adapts the model to new domains. FIG. 25 shows a few qualitative results before and after fine-tuning.

Model	KITTI	ADE	Rooftop	Card.MR	ssTEM
Square Box (Perfect)	-	69.35	62.11	79.11	66.53
Ellipse (Perfect)	-	69.53	66.82	92.44	71.32
Polygon-RNN++ (BS)	83.14	71.82	65.67	80.63	53.12
PSP-DeepLab	83.35	72.70	57.91	74.11	47.65
Spline-GCN	84.09	72.94	68.33	78.54	58.46
+ finetune	84.81	77.35	78.21	91.33	-
Polygon-GCN	83.66	72.31	66.78	81.55	60.91
+ finetune	84.71	77.41	75.56	90.91	-

Table 11

[00190] Thus, the Curve-GCN model may both provide an increase in speed over previous models, as well as enabling interactive corrections which are restricted to being local in effect, thereby providing more control to the annotator.

[00191] Alternative Method Embodiment

[00192] As shown in FIG. 27, an embodiment of the alternative method 27000 may comprise steps of receiving an image depicting an object, the image comprising an n-dimensional array-like structure 27001, and generating one or more image features using a CNN encoder implemented on one or more computers 27002, initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along an ellipse centered in the image 27003, predicting a location shift for each node simultaneously using a Graph Convolutional Network (GCN) 27004, iterating predictions through the GCN for each node, each iteration defining a new location shift for each node based on node locations for each node from the previous iteration 27005, and producing an object annotation based on a final iteration, wherein the object is parametrized with one of polygons and splines 27006.

[00193] According to an embodiment, the method includes training and testing with a fixed number of iterations. That number can be arbitrarily changed according to the user's choice.

[00194] Alternative System Embodiment

[00195] An embodiment of a system to carry out the alternative method is shown in FIG. 28. In sample system 28000, an input unit 28010 receives an image depicting an object, the image comprising an n-dimensional array-like structure at a computer which includes a system

processor 28020 which may comprise processing and other necessary elements distributed across one or more computers for processing the input. At system processor 28020, a CNN encoder 28021 generates one or more image features and initializes a set of N nodes from the set of image features where the set of N nodes forms a closed curve along an ellipse centered in the image and predicts a location shift for each node simultaneously using a GCN 28023. Predictions are iterated through the GCN for each node where each iteration is a separate set of layers of the neural network and each iteration defines a new location shift for each node based on node locations for each node from the previous iteration, eventually producing an object annotation selected by output selector 28024 based on a final iteration, wherein the object is parametrized with one of polygons and splines and provided to output or application unit 28030 such as a computer monitor or other display or an output unit for sending annotated objects to another system for a particular application or use.

[00196] Potential use cases for this alternative method and system embodiment may include the delineation of 3D objects from multiple views to generate coarse mesh annotation. Further, household objects or other specified relevant objects (clothing, persons, etc.) may be annotated for robotic perception or other person-related software. Other applications may include completing partially drawn annotations and object selection for photo editing software.

[00197] While various elements, blocks, or units are depicted or described as being either independent or as being components of other elements, blocks, or units, in other embodiments other arrangements of elements, blocks or units may be employed.

[00198] Various embodiments of the invention have been described in detail. Since changes in and or additions to the above-described best mode may be made without departing from the nature, spirit or scope of the invention, the invention is not to be limited to those details but only by the appended claims. Section headings herein are provided as organizational cues. These headings shall not limit or characterize the invention set out in the appended claims.

What is claimed is:

1. A method of annotating an object, comprising:
 - receiving an image depicting an object, the image comprising an n-dimensional array-like data structure;
 - generating a set of image features using a CNN encoder implemented on one or more computers;
 - predicting a set of vertex predictions using the set of image features;
 - producing a set of polygon predictions of the object using a recurrent decoder that exploits the set of vertex predictions and the set of image features, the recurrent decoder implemented on one or more computers; and
 - selecting a polygon object annotation from the set of polygon predictions.
2. The method of claim 1, wherein the selecting of a polygon object annotation is performed using an evaluator network.
3. The method of claim 1, further comprising generating a higher resolution polygon from the polygon object annotation using a graph neural network implemented on one or more computers.
4. The method of claim 3, wherein generating a higher resolution polygon comprises:
 - adding a set of supplementary vertex predictions to a set of primary vertex predictions defining the polygon objection annotation;
 - defining a propagation model; and
 - applying the propagation model to adjust the position of the vertices of the set of supplementary vertex predictions and the vertices of the set of primary vertex predictions.

5. The method of claim 3, wherein the recurrent decoder applies an attention unit at each time step.
6. The method of claim 1, wherein producing a set of polygon representations of an object using a recurrent decoder includes producing a weighted feature map.
7. The method of claim 2, wherein the evaluator network predicts an Intersection over Union (IoU) of each polygon prediction of the set of polygon predictions using gamma testing, and the polygon object annotation is the polygon prediction of the set of polygon predictions having the maximum IoU.
8. The method of claim 2, wherein producing each polygon prediction of the set of polygon predictions includes a series of timesteps to produce a set of vertex predictions defining the polygon prediction and the evaluator network is applied at each timestep of the series of timesteps.
9. The method of claim 1, wherein the image is selected from one of: an RGB image, a thermal image, a depth image, and a hyperspectral image.
10. The method of claim 1, wherein the recurrent decoder is one of: a recurrent neural network, a graph neural network and a gated graph neural network.
11. The method of claim 1, wherein the set of polygon predictions comprises one or more human corrections to the set of vertex predictions.
12. The method of claim 1, wherein the polygon object annotation is selected by selecting each vertex according to a highest log probability for the vertex.
13. A system for object annotation, comprising:
 - a CNN encoder implemented by one or more computers for generating image features from a received image, the image comprising an n-dimensional array-like data structure, and the image features for predicting one or more vertexes of an object annotation;

a recurrent decoder for generating a set of polygon predictions of an object in the received image, the recurrent decoder implemented by one or more computers; and
a selector for selecting a polygon object annotation from the set of polygon predictions.

14. The system of claim 13, wherein the recurrent decoder includes a Recurrent Neural Network (RNN).
15. The system of claim 13, wherein the recurrent decoder includes an attention unit at each time step to produce each polygon of the set of polygon predictions one vertex at a time.
16. The system of claim 13, further comprising a gated graph neural network for generating a higher resolution polygon from the selected polygon prediction.
17. The system of claim 16, wherein the gated graph neural network includes a propagation block and an output block.
18. The system of claim 13, further comprising an application unit for receiving a resultant object annotation.
19. The system of claim 18, wherein the application unit is one of an autonomous driving system, a medical imaging system, and a mapping system.
20. The system of claim 13, wherein the CNN encoder includes a skip layer architecture.
21. The system of claim 13, further comprising a human input interface for receiving one or more human corrections to the set of polygon predictions.
22. The system of claim 13, wherein the selector selects the polygon object annotation based on selecting each vertex according to a highest log probability for the vertex.
23. A method of training an object annotation system having a CNN encoder, an RNN recurrent decoder and an evaluator network, comprising:
receiving a training dataset;

initiating a training sequence for setting one or more weight matrices of the object annotation system using managed learning environment training; and

fine-tuning the one or more weight matrices of the object annotation system to produce a trained object annotation system.

24. The method of claim 23, wherein the method of training further comprises providing simulated human correction.
25. The method of claim 23, wherein the method of training further comprises providing human correction.
26. The method of claim 23, wherein the fine-tuning comprises using reinforcement learning.
27. The method of claim 23, wherein the evaluator network is trained separately from the rest of the object annotation system after the fine-tuning using reinforcement learning has converged.
28. The method of claim 23, further including producing an object annotation prediction for an image of the training dataset.
29. The method of claim 28, further including submitting the object annotation prediction for human correction.
30. The method of claim 29, further including producing a human correction of the object annotation prediction and feeding the human correction back into the object annotation system to further train the object annotation system.
31. The method of claim 23, further including fine-tuning the object annotation system using online fine-tuning.
32. A method of annotating an object, comprising:

receiving an image depicting an object, the image comprising an n-dimensional array-like data structure;

generating a set of image features using a CNN encoder implemented on one or more computers;

initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image;

predicting a location shift for each node simultaneously using a Graph Neural Network (GNN);

iterating predictions through the GNN for each node, each iteration defining a new location shift for each node based on node locations for each node from the previous iteration; and

producing an object annotation based on a final iteration;

wherein the object is parametrized with one of polygons and splines.

33. The method of claim 32, further comprising applying one or more simulated human corrections to one or more nodes during the iteration step.
34. The method of claim 32, further comprising applying one or more human corrections to one or more nodes during the iteration step.
35. The method of claim 32, wherein the GNN is trained via point matching loss.
36. The method of claim 35, wherein the GNN is further trained and fine-tuned via differentiable accuracy loss.
37. The method of claim 32, wherein a centripetal Catmull-Ross spline is applied for splines.
38. The method of claim 32, wherein the image is selected from one of: an RGB image, a thermal image, a depth image, and a hyperspectral image.
39. The method of claim 32, wherein the final iteration is based on running a pre-selected number of iterations.
40. The method of claim 32, wherein the GNN is a Graph Convolutional Network.

41. A system for object annotation, comprising:
- a CNN encoder implemented by one or more computers for generating a set of image features from a received image, the image comprising an n-dimensional array-like data structure, and for initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image;
- a Graph Neural Network (GNN) implemented by one or more computers for predicting a location shift for each node simultaneously and iterating predictions through the GNN for each node, each iteration defining a new location shift for each node based on node locations for each node from the previous iteration; and
- an output selector for producing an output based on a final iteration from the GNN.
42. The system of claim 41, further comprising a human input interface for receiving one or more human corrections to one or more nodes during the iteration.
43. The system of claim 41, further comprising a training system implemented by one or more computers for training the GNN.
44. The system of claim 43, wherein the training system applies a point matching loss training method.
45. The system of claim 43, wherein the training system applies a differentiable accuracy loss method.
46. The system of claim 43, wherein the training system applies simulated human corrections.
47. The system of claim 43, wherein the training system applies human corrections.
48. The system of claim 43, wherein the final iteration is based on running a pre-selected number of iterations.
49. The system of claim 41, wherein the GNN is a Graph Convolutional Network.

50. A non-transient computer-readable medium comprising instructions for a method of annotating an object, the method comprising:

receiving an image depicting an object, the image comprising an n-dimensional array-like data structure;

generating a set of image features using a CNN encoder implemented on one or more computers;

predicting a set of first vertex predictions using the set of image features;

producing a set of polygon predictions of the object using a recurrent decoder that exploits the set of first vertex predictions and the set of image features, the recurrent decoder implemented on one or more computers; and

selecting a polygon object annotation from the set of polygon predictions.

51. The non-transient computer-readable medium of claim 50, wherein the selecting of a polygon object annotation is performed using an evaluator network.

52. The non-transient computer-readable medium of claim 50, further comprising instructions for generating a higher resolution polygon from the polygon object annotation using a graph neural network implemented on one or more computers.

53. The non-transient computer-readable medium of claim 52, wherein generating a higher resolution polygon comprises:

adding a set of supplementary vertex predictions to a set of primary vertex predictions defining the polygon objection annotation;

defining a propagation model; and

applying the propagation model to adjust the position of the vertices of the set of supplementary vertex predictions and the vertices of the set of primary vertex predictions.

54. The non-transient computer-readable medium of claim 52, wherein the recurrent decoder applies an attention unit at each time step.
55. The non-transient computer-readable medium of claim 50, wherein producing a set of polygon representations of an object using a recurrent decoder includes producing a weighted feature map.
56. The non-transient computer-readable medium of claim 51, wherein the evaluator network predicts an Intersection over Union (IoU) of each polygon prediction of the set of polygon predictions using gamma testing, and the polygon object annotation is the polygon prediction of the set of polygon predictions having the maximum IoU.
57. The non-transient computer-readable medium of claim 51, wherein producing each polygon prediction of the set of polygon predictions includes a series of timesteps to produce a set of vertex predictions defining the polygon prediction and the evaluator network is applied at each timestep of the series of timesteps.
58. The non-transient computer-readable medium of claim 50, wherein the image is selected from one of: an RGB image, a thermal image, a depth image, and a hyperspectral image.
59. The non-transient computer-readable medium of claim 50, wherein the recurrent decoder is one of: a recurrent neural network, a graph neural network and a gated graph neural network.
60. The non-transient computer-readable medium of claim 50, wherein the set of polygon predictions comprises one or more human corrections to set of first vertex predictions.
61. The non-transient computer-readable medium of claim 50, wherein the polygon object annotation is selected by selecting each vertex according to a highest log probability for the vertex.
62. A non-transient computer-readable medium comprising instructions for a method of annotating an object, the method comprising:

receiving an image depicting an object, the image comprising an n-dimensional array-like data structure;

generating a set of image features using a CNN encoder implemented on one or more computers;

initializing a set of N nodes from the set of image features, the set of N nodes forming a closed curve along a circle centered in the image;

predicting a location shift for each node simultaneously using a Graph Neural Network (GNN);

iterating predictions through the GNN for each node, each iteration defining a new location shift for each node based on node locations for each node from the previous iteration; and

producing an object annotation based on a final iteration;

wherein the object is parametrized with one of polygons and splines.

63. The non-transient computer-readable medium of claim 62, further comprising applying one or more simulated human corrections to one or more nodes during the iteration step.
64. The non-transient computer-readable medium of claim 62, further comprising applying one or more human corrections to one or more nodes during the iteration step.
65. The non-transient computer-readable medium of claim 62, wherein the GNN is trained via point matching loss.
66. The non-transient computer-readable medium of claim 65, wherein the GNN is further trained and fine-tuned via differentiable accuracy loss.
67. The non-transient computer-readable medium of claim 62, wherein a centripetal Catmull-Ross spline is applied for splines.

68. The non-transient computer-readable medium of claim 62, wherein the image is selected from one of: an RGB image, a thermal image, a depth image, and a hyperspectral image.
69. The non-transient computer-readable medium of claim 62, wherein the final iteration is based on running a pre-selected number of iterations.
70. The non-transient computer-readable medium of claim 62, wherein the GNN is a Graph Convolutional Network.
71. A non-transient computer-readable medium comprising instructions for a method of training an object annotation system having a CNN encoder, an RNN recurrent decoder and an evaluator network, the method comprising:
- receiving a training dataset;
 - initiating a training sequence for setting one or more weight matrices of the object annotation system using managed learning environment training; and
 - fine-tuning the one or more weight matrices of the object annotation system to produce a trained object annotation system.
72. The non-transient computer-readable medium of claim 71, wherein the method of training further comprises providing simulated human correction.
73. The non-transient computer-readable medium of claim 71, wherein the method of training further comprises providing human correction.
74. The non-transient computer-readable medium of claim 71, wherein the fine-tuning comprises using reinforcement learning.
75. The non-transient computer-readable medium of claim 71, wherein the evaluator network is trained separately from the rest of the object annotation system after the fine-tuning using reinforcement learning has converged.
76. The non-transient computer-readable medium of claim 71, further including producing an object annotation prediction for an image of the training dataset.

77. The non-transient computer-readable medium of claim 76, further including submitting the object annotation prediction for human correction.
78. The non-transient computer-readable medium of claim 77, further including producing a human correction of the object annotation prediction and feeding the human correction back into the object annotation system to further train the object annotation system.
79. The non-transient computer-readable medium of claim 71, further including fine-tuning the object annotation system using online fine-tuning.

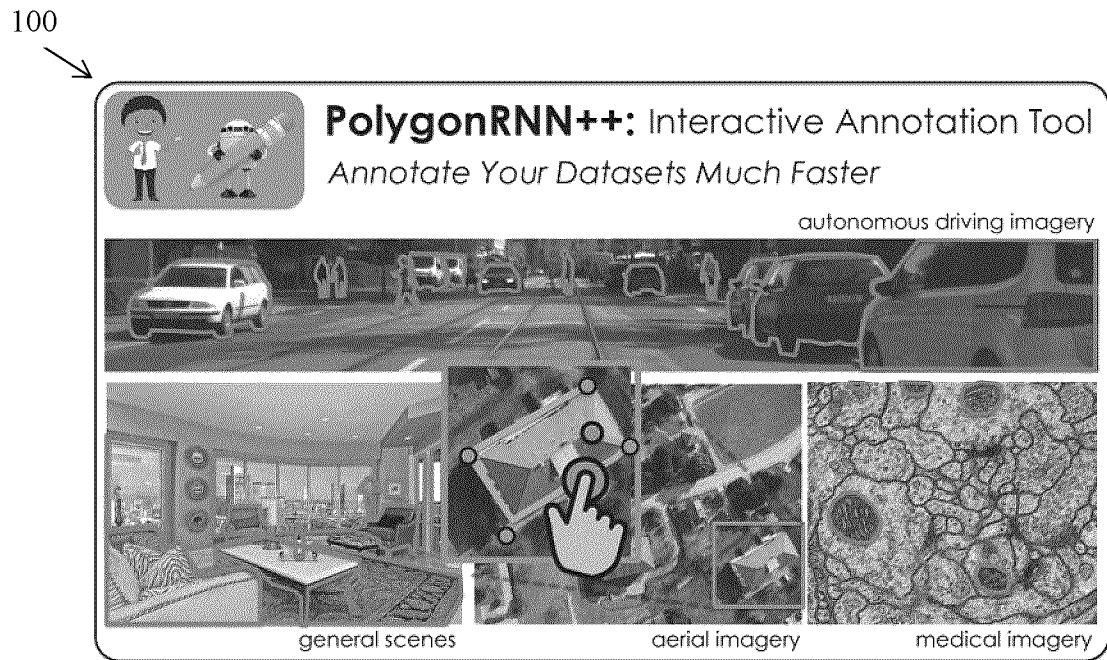


FIG. 1

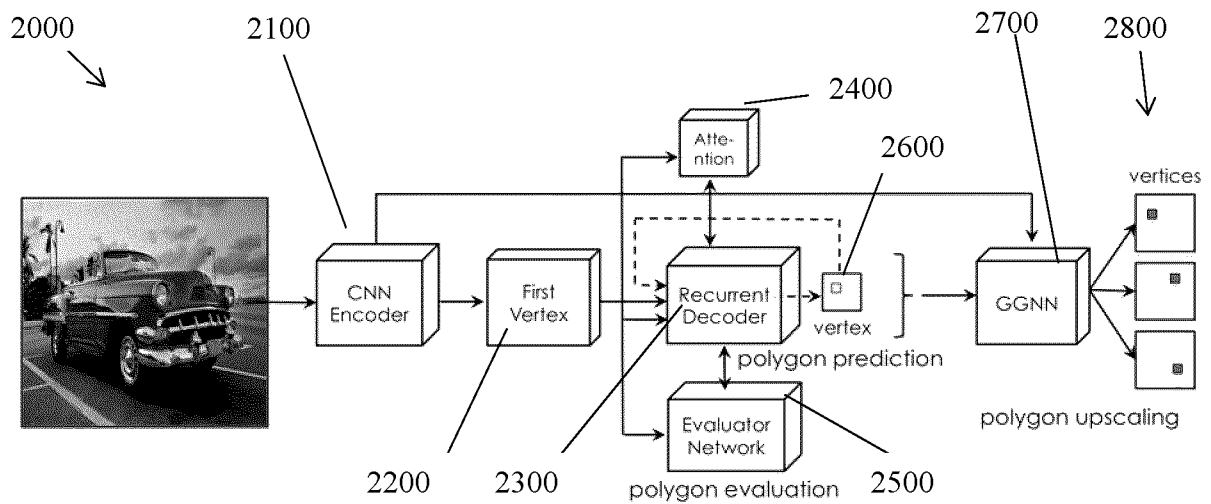


FIG. 2

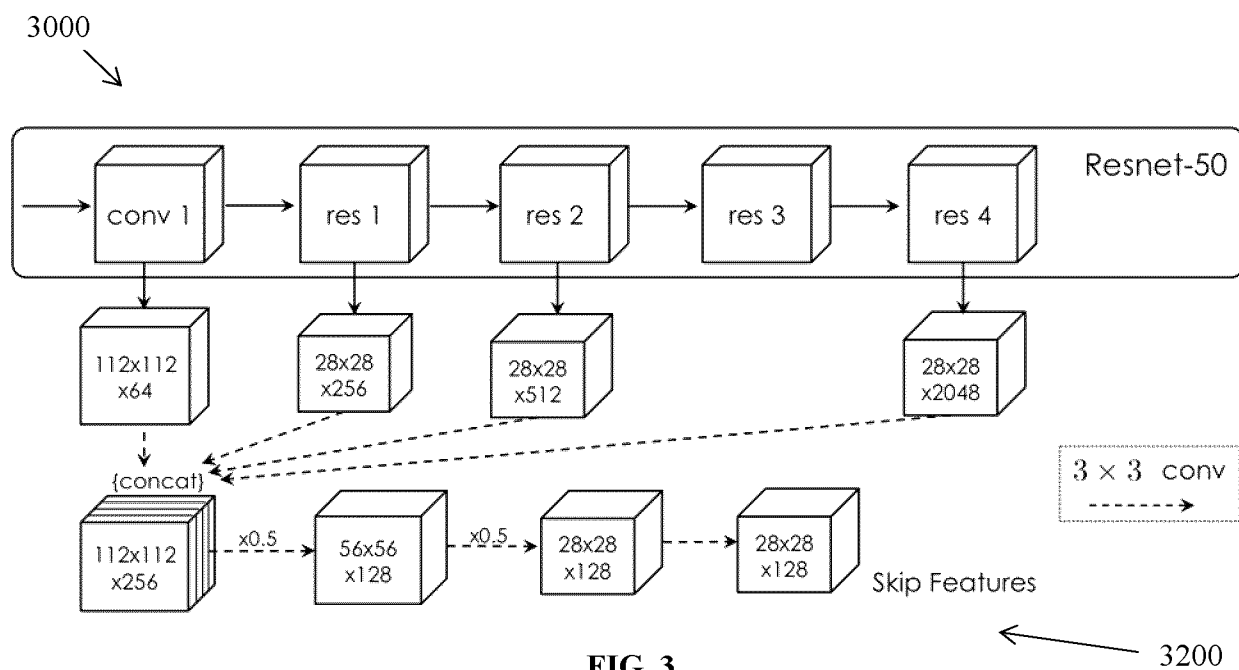


FIG. 3

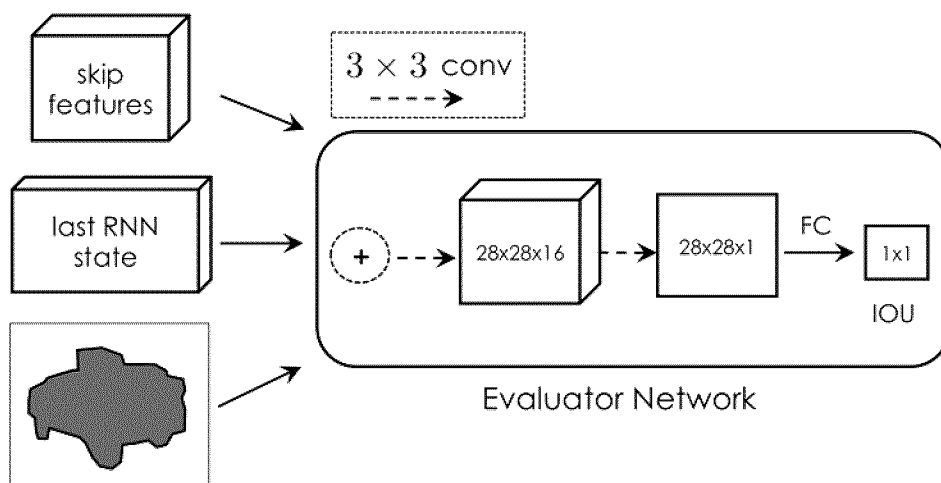


FIG. 4

3/18

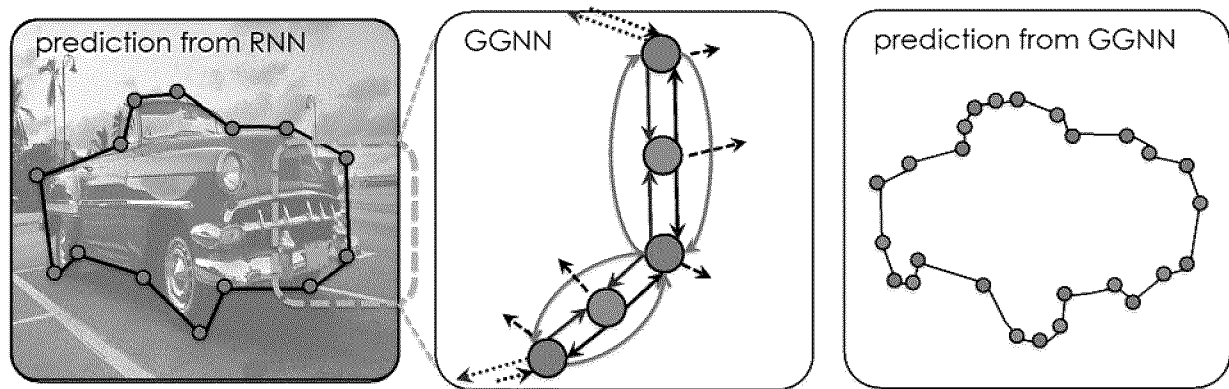


FIG. 5

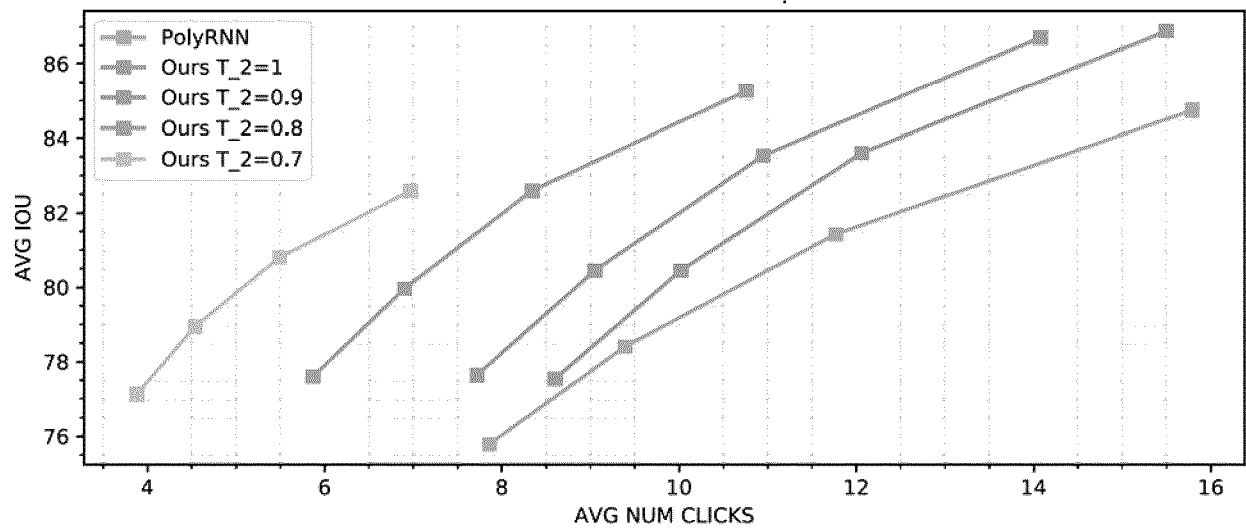


FIG. 6

4/18

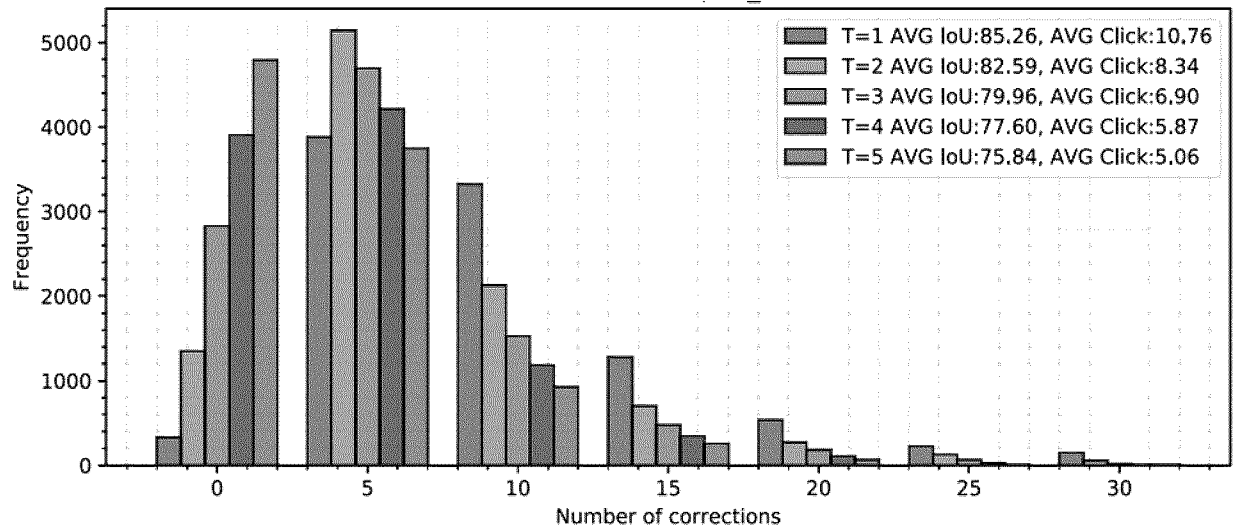


FIG. 7

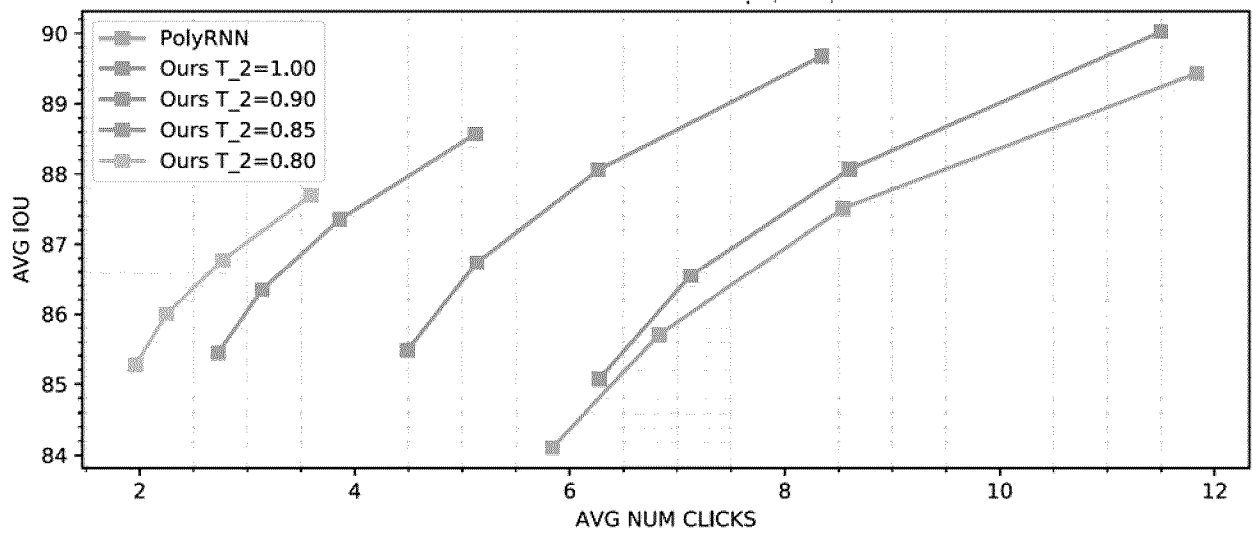


FIG. 8

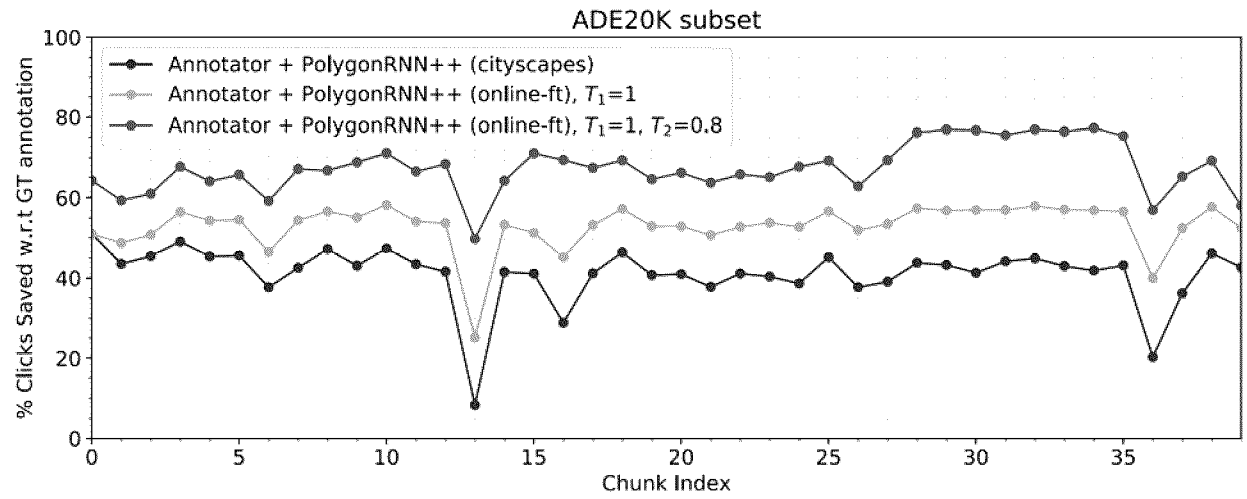


FIG. 9A

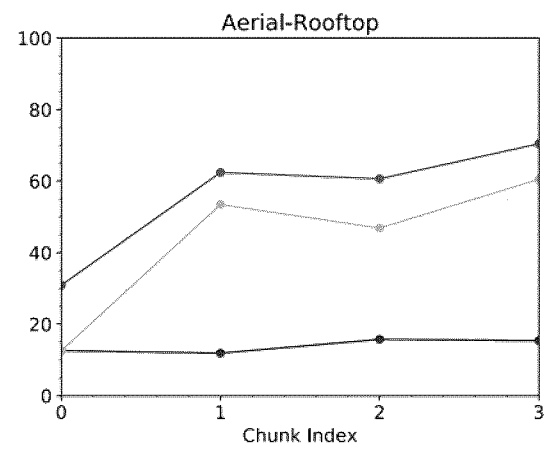


FIG. 9B

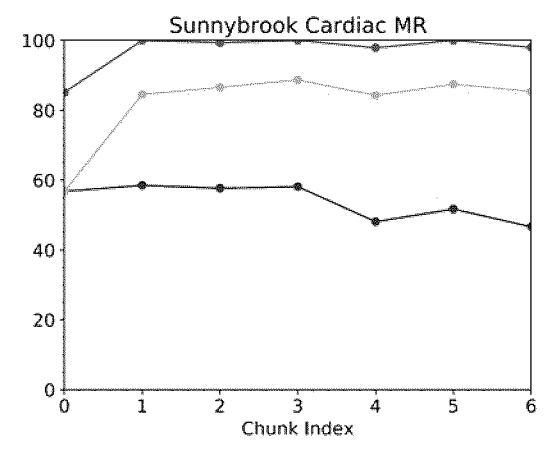


FIG. 9C

6/18

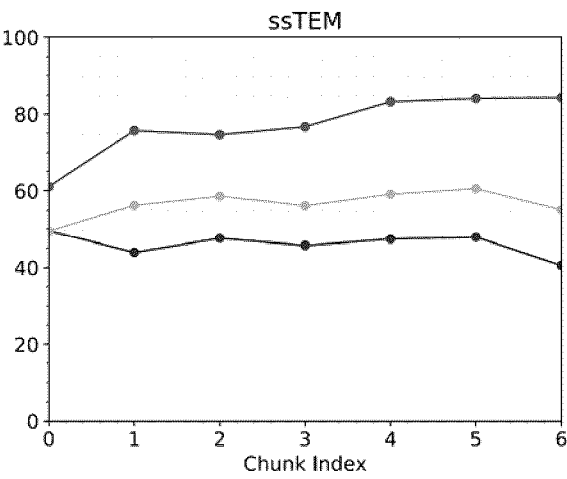


FIG. 9D



FIG. 10

7/18

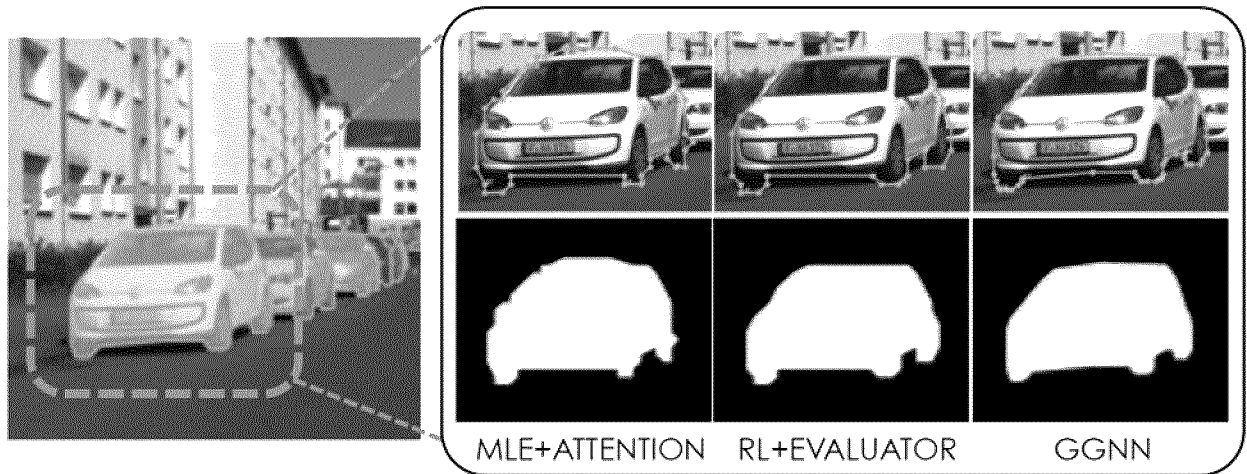


FIG. 11

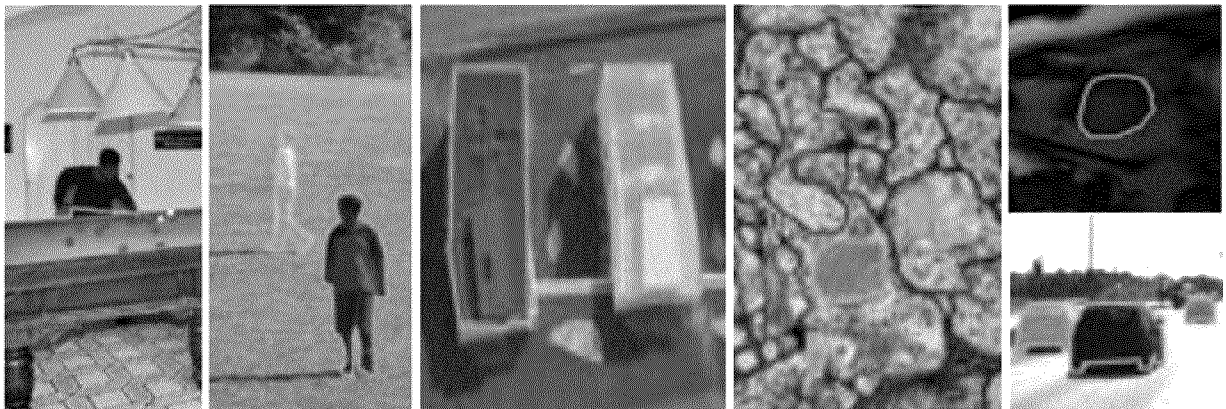


FIG. 12

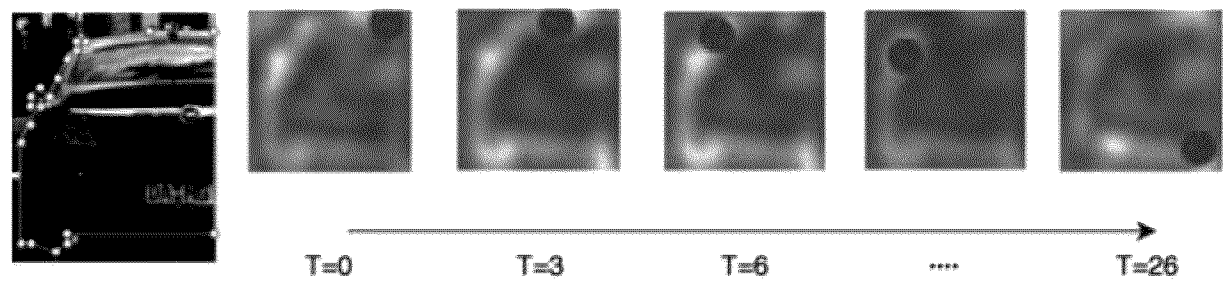


FIG. 13

9/18

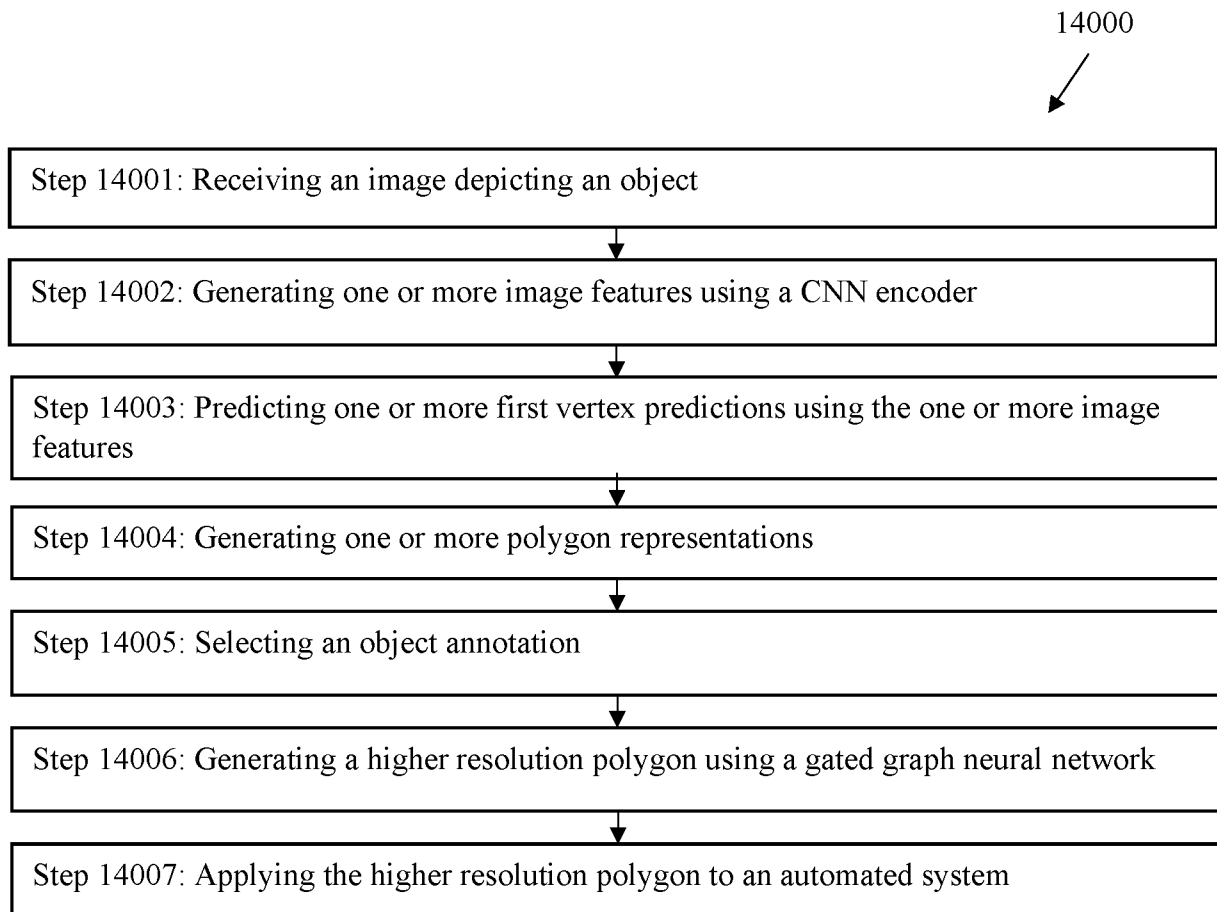


FIG. 14

10/18

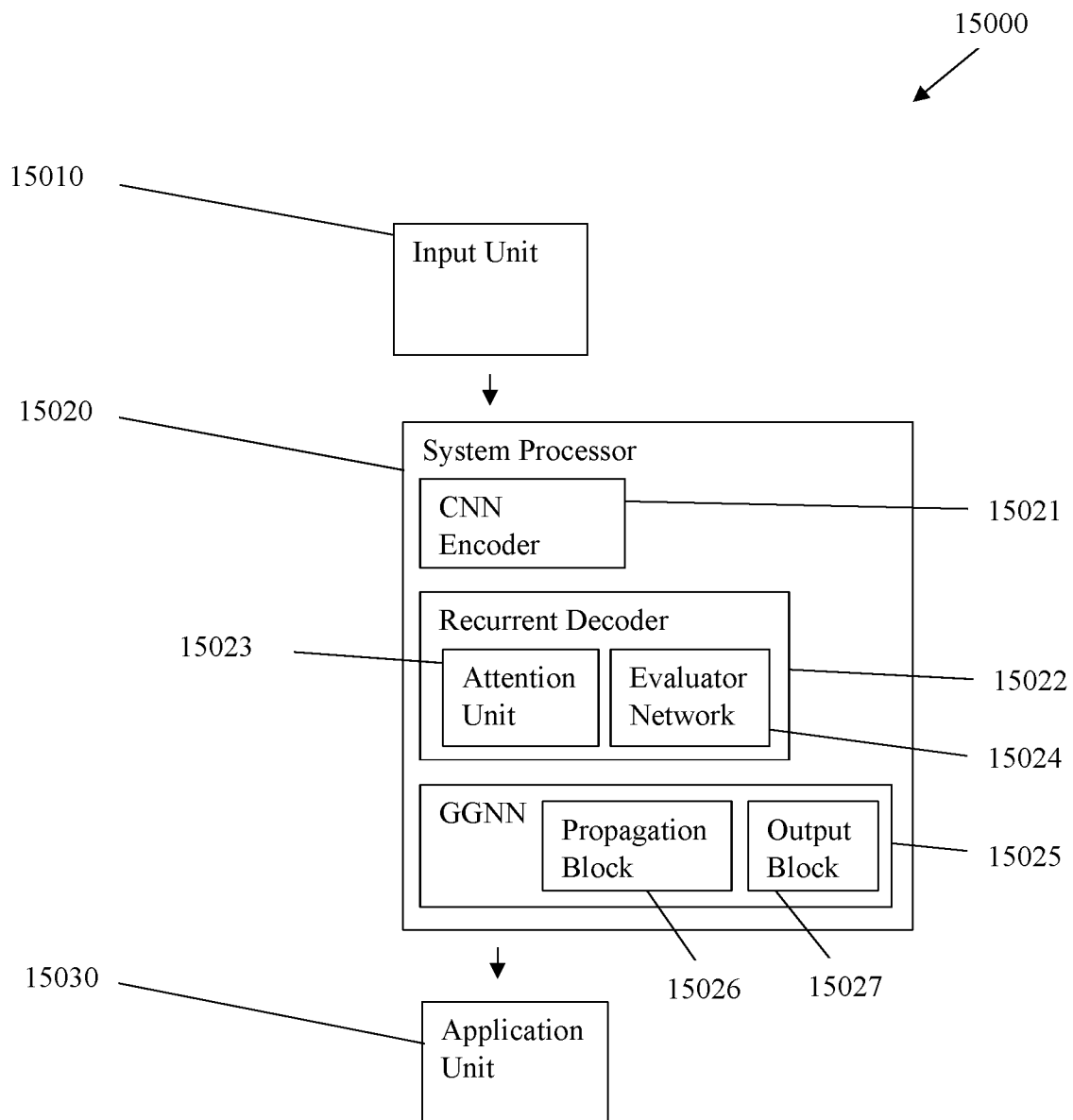


FIG. 15

11/18

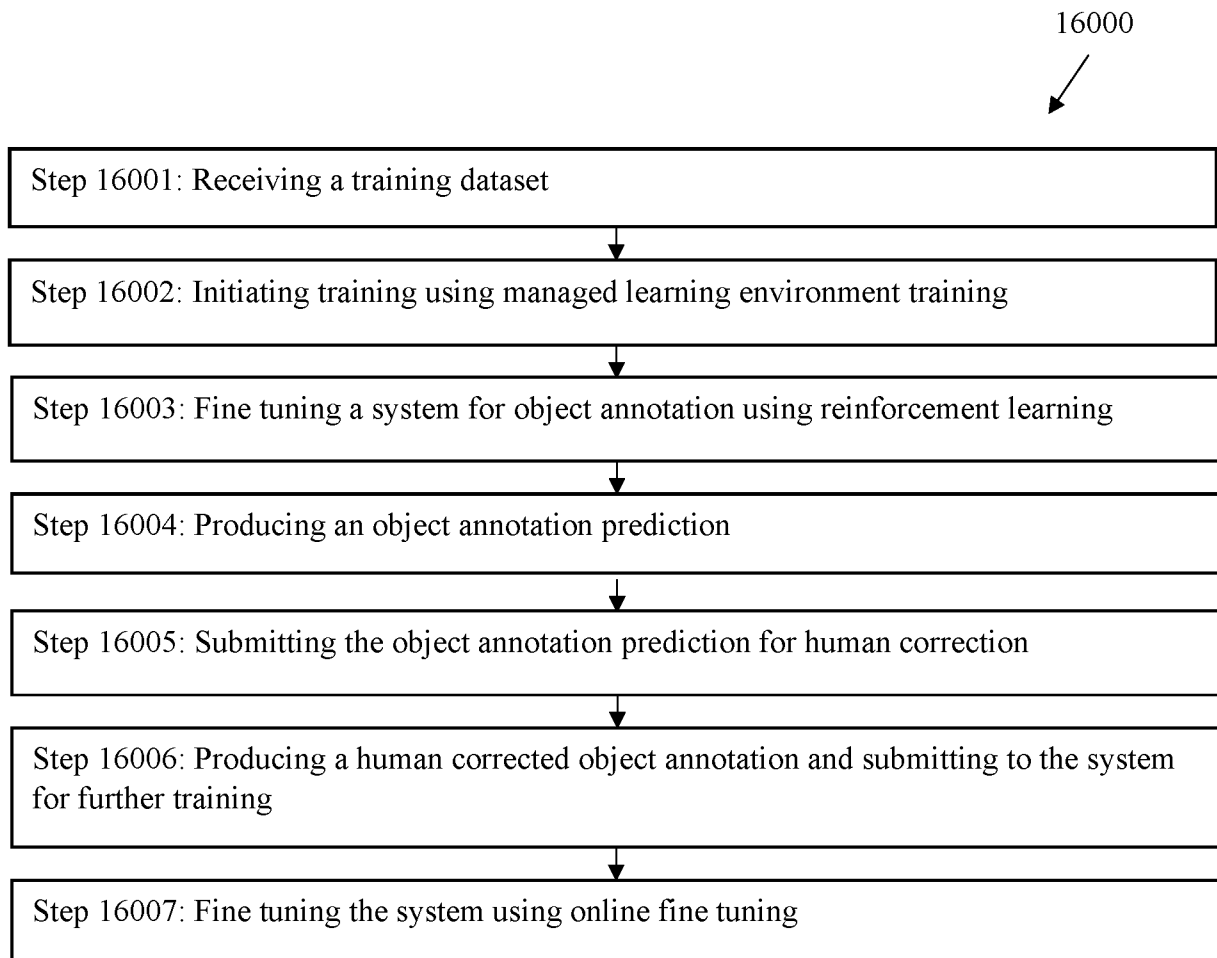


FIG. 16

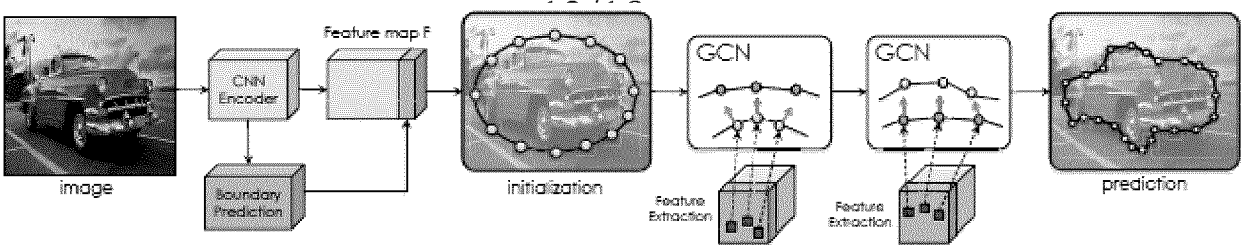


FIG. 17

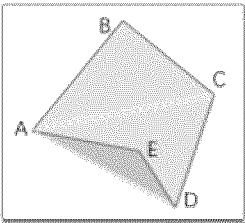


FIG. 18

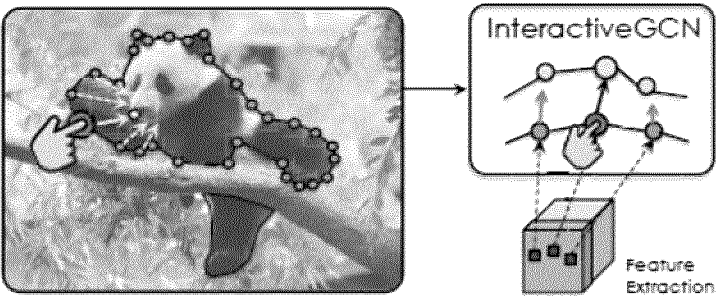


FIG. 19



FIG. 20

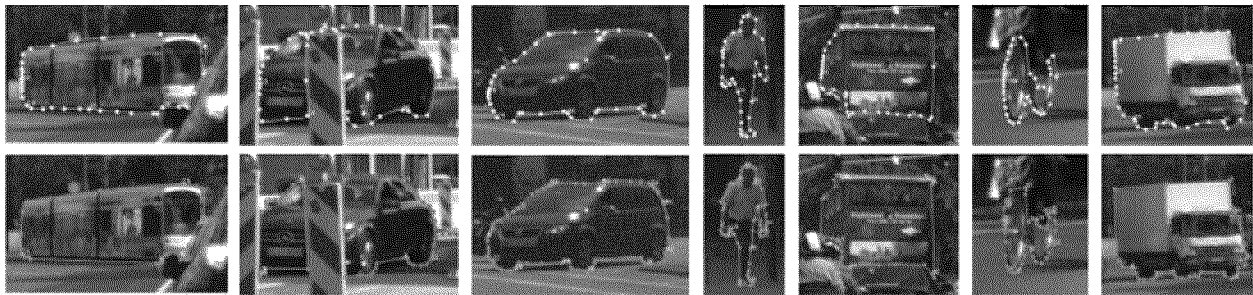


FIG. 21



FIG. 22

14/18

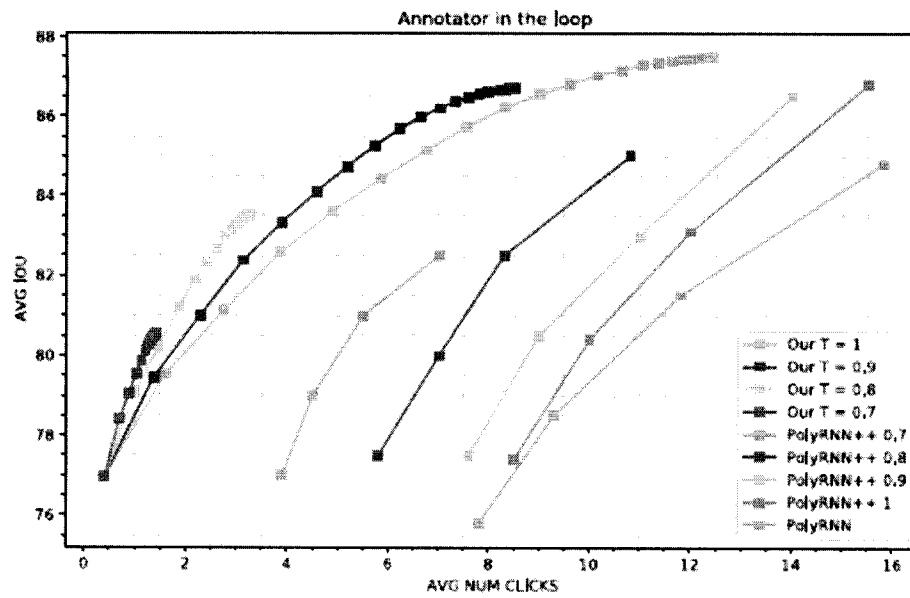


FIG. 23A

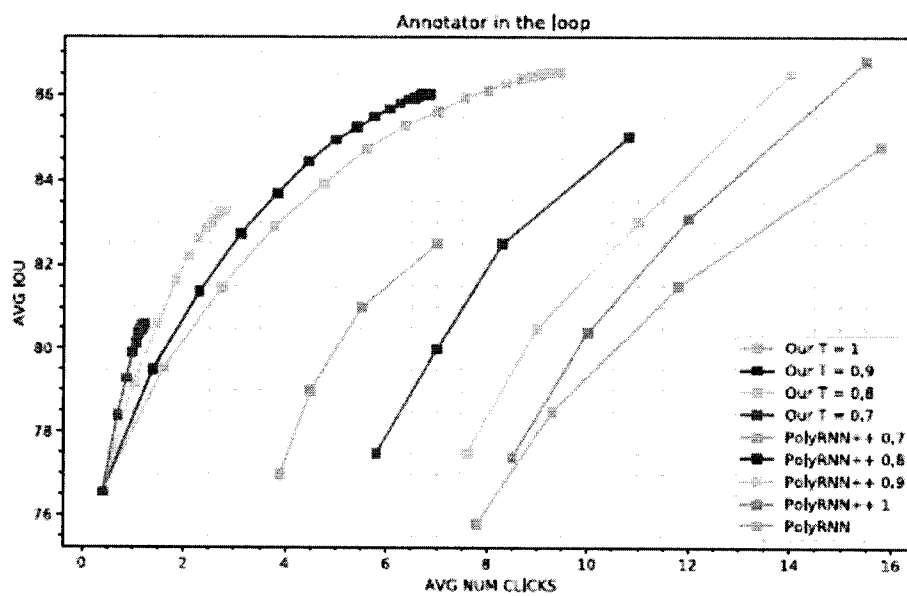


FIG. 23B

15/18

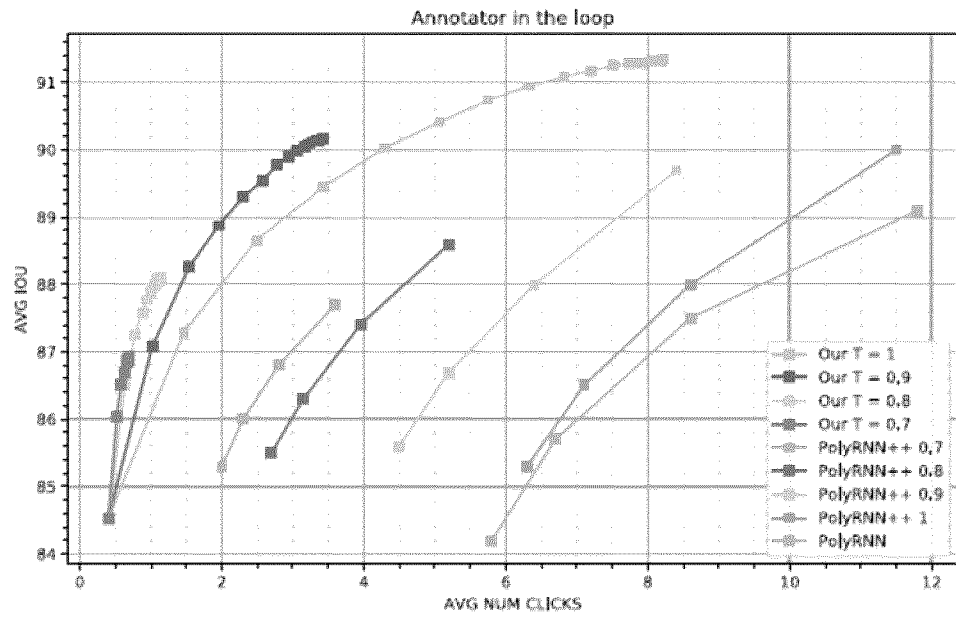


FIG. 24

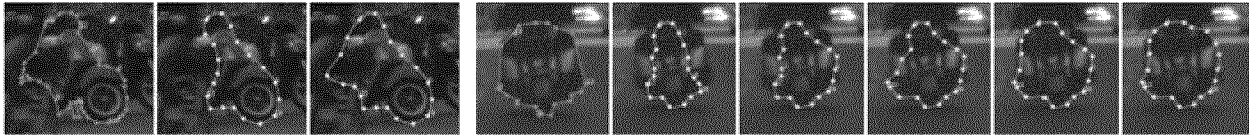


FIG. 25

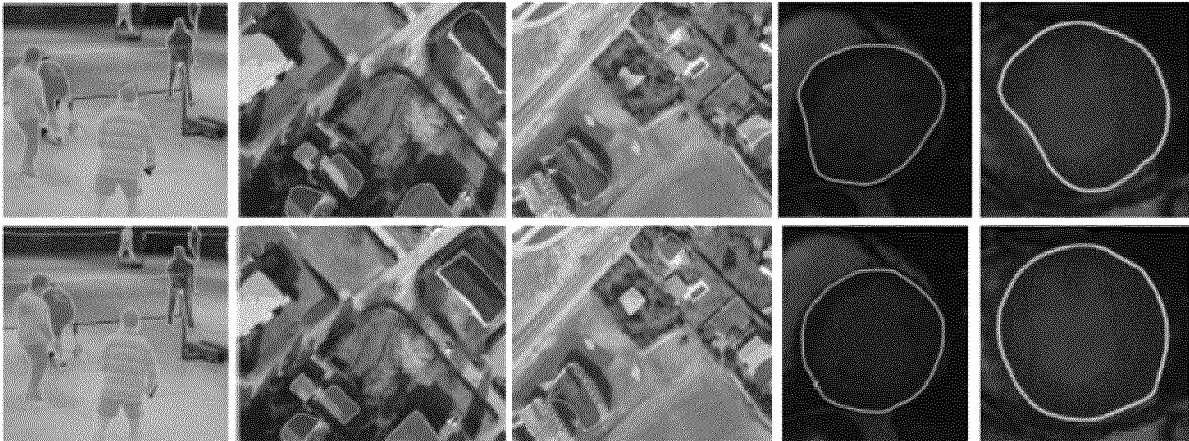


FIG. 26

17/18

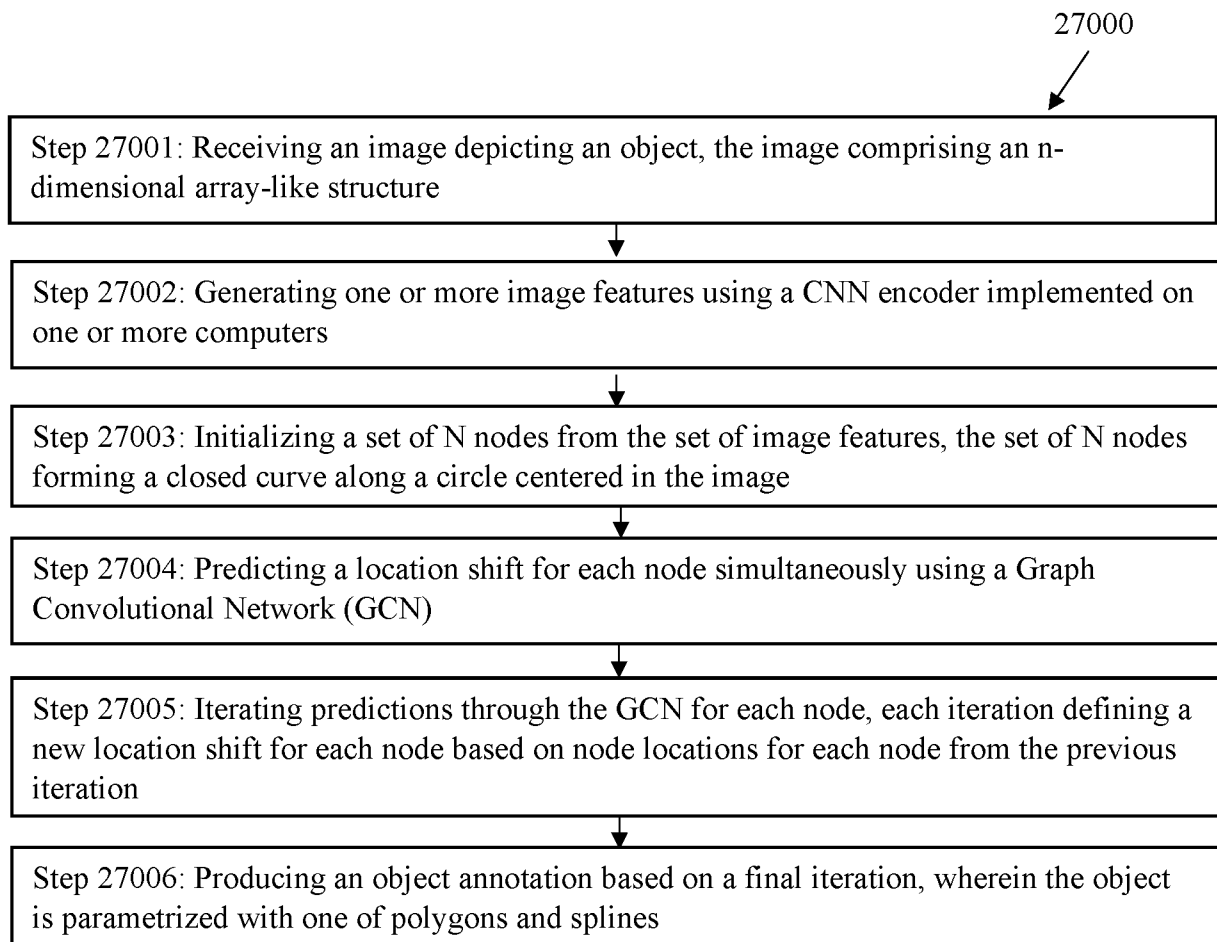


FIG. 27

18/18

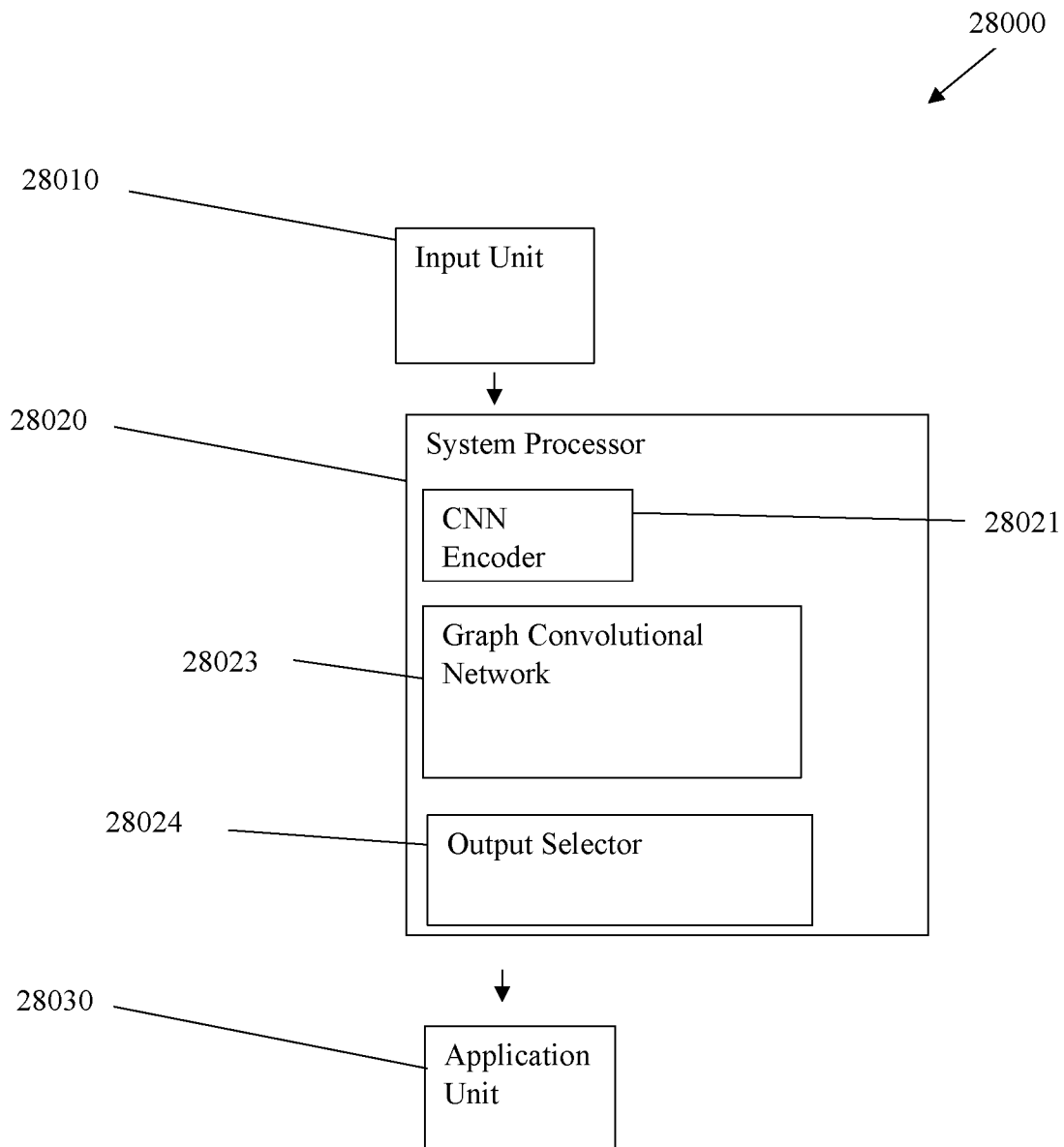


FIG. 28

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CA2019/050362

A. CLASSIFICATION OF SUBJECT MATTER

IPC: **G06K 9/46** (2006.01), **G06N 3/02** (2006.01), **G06N 3/08** (2006.01), **G06T 1/40** (2006.01),
G06T 11/60 (2006.01), **G06T 7/10** (2017.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06K 9/46 (2006.01), **G06K 9/66** (2006.01), **G06N 3/02** (2006.01), **G06N 3/08** (2006.01), **G06T 1/40** (2006.01),
G06T 11/60 (2006.01), **G06T 7/10** (2017.01)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

- N / A

Electronic database(s) consulted during the international search (name of database(s) and, where practicable, search terms used)

Databases: Intellect (CIPO Database), IEEE Xplore, Questel Orbit, Google, Google scholar, Google Patents

Keywords: graph, neural, node, object, outline, polygon, predict, recognition, refine, CNN, RNN, GNN, image, shift, training, weight, vertex, vertices, "fine tune", annotat(e, ion), bounding, contour, curve, encoder

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	CASTREJON et al., Annotating Object Instances with a Polygon-RNN, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 18 April 2017 (18-04-2017), pp. 5230-5238, ISSN 1063-6919 [online] [retrieved on 14 May 2019 (14-05-2019)].	1-22 and 50-61
Y	Retrieved from the Internet: http://openaccess.thecvf.com/content_cvpr_2017/papers/Castrejon_Annotating_Object_Instances_CVPR_2017_paper.pdf * whole document	28-30, 32-49, 62-70 and 76-78
X	WO2015/011688A2 (LILLICRAP et al.) 29 January 2015 (29-01-2015), Paragraphs [4]-[6], [36] and [38]	23-26 and 71-74
Y		27-31 and 75-79

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:	"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"A" document defining the general state of the art which is not considered to be of particular relevance	"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"E" earlier application or patent but published on or after the international filing date	"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)	"&" document member of the same patent family
"O" document referring to an oral disclosure, use, exhibition or other means	
"P" document published prior to the international filing date but later than the priority date claimed	

Date of the actual completion of the international search
13 May 2019 (13-05-2019)

Date of mailing of the international search report
22 May 2019 (22-05-2019)

Name and mailing address of the ISA/CA
Canadian Intellectual Property Office
Place du Portage I, C114 - 1st Floor, Box PCT
50 Victoria Street
Gatineau, Quebec K1A 0C9
Facsimile No.: 819-953-2476

Authorized officer

Giles Babin (819) 639-2193

INTERNATIONAL SEARCH REPORT

International application No.
PCT/CA2019/050362

Box No. II Observations where certain claims were found unsearchable (Continuation of item 2 of the first sheet)

This international search report has not been established in respect of certain claims under Article 17(2)(a) for the following reasons:

1. ☐ Claim Nos.:
because they relate to subject matter not required to be searched by this Authority, namely:

2. ☐ Claim Nos.:
because they relate to parts of the international application that do not comply with the prescribed requirements to such an extent that no meaningful international search can be carried out, specifically:

3. ☐ Claim Nos.:
because they are dependent claims and are not drafted in accordance with the second and third sentences of Rule 6.4(a).

Box No. III Observations where unity of invention is lacking (Continuation of item 3 of first sheet)

This International Searching Authority found multiple inventions in this international application, as follows:

Group A - Claims 1-22 and 50-61 are directed to a system and method for annotating an object in an image wherein the image is analysed by a CNN and a set of vertex and polygon predictions is generated. These are then used to select object annotations;

Group B - Claims 23-31 and 71-79 are directed to a method of training an object annotation system; and

Group C - Claims 32-49 and 62-70 are directed to a system and method for annotating an object in an image wherein the image is analysed by a CNN and a set of features is produced which are then used by a GNN to determine a set of node location shifts in multiple iterations. This final iteration is used to generate the object annotations.

1. ☒ As all required additional search fees were timely paid by the applicant, this international search report covers all searchable claims.
2. ☐ As all searchable claims could be searched without effort justifying additional fees, this Authority did not invite payment of additional fees.
3. ☐ As only some of the required additional search fees were timely paid by the applicant, this international search report covers only those claims for which fees were paid, specifically claim Nos.:

4. ☐ No required additional search fees were timely paid by the applicant. Consequently, this international search report is restricted to the invention first mentioned in the claims; it is covered by claim Nos.:

Remark on Protest

- ☐ The additional search fees were accompanied by the applicant's protest and, where applicable, the payment of a protest fee.
- ☐ The additional search fees were accompanied by the applicant's protest but the applicable protest fee was not paid within the time limit specified in the invitation.
- ☒ No protest accompanied the payment of additional search fees.

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CA2019/050362

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	LIU et al., Semantic Regularisation for Recurrent Image Annotation, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 26 July 2017 (26-07-2017), pp. 2872-2880, ISSN 1063-6919 [online] [retrieved on 14 May 2019 (14-05-2019)]. Retrieved from the Internet: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8099926 Section 3.3; "Training and inference"	27 and 75
Y	CN101814149B (TANG et al.) 25 August 2010 (25-08-2010), Paragraph [0037]	31 and 79
Y	QUEK et al., Structural Image Classification with Graph Neural Networks, 2011 International Conference on Digital Image Computing: Techniques and Applications, 8 December 2011 (08-12-2011), pp. 416-421, [online] [retrieved on 14 May 2019 (14-05-2019)]. Retrieved from the Internet: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6128754 * whole document	32-49 and 62-70

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CA2019/050362

Patent Document Cited in Search Report	Publication Date	Patent Family Member(s)	Publication Date
WO2015011688A2	29 January 2015 (29-01-2015)	WO2015011688A2 WO2015011688A3 EP3025277A2 GB201402736D0 US2016162781A1	29 January 2015 (29-01-2015) 14 May 2015 (14-05-2015) 01 June 2016 (01-06-2016) 02 April 2014 (02-04-2014) 09 June 2016 (09-06-2016)
CN101814149A	25 August 2010 (25-08-2010)	CN101814149A CN101814149B	25 August 2010 (25-08-2010) 25 January 2012 (25-01-2012)